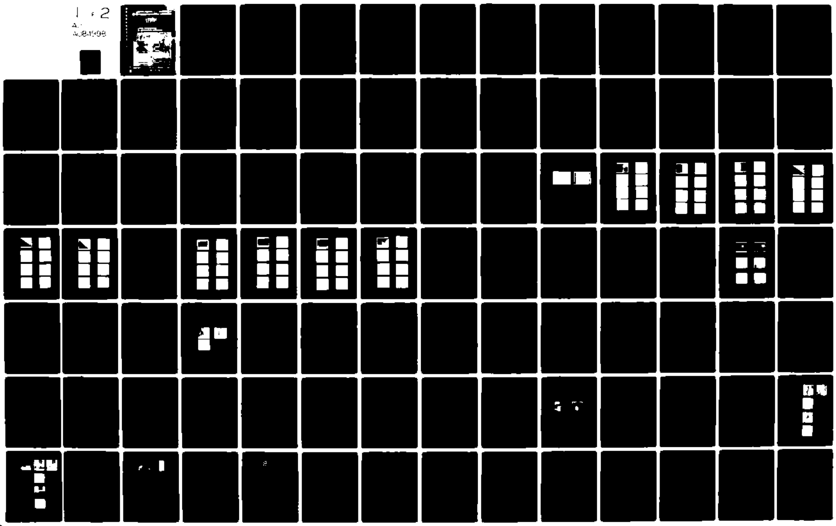
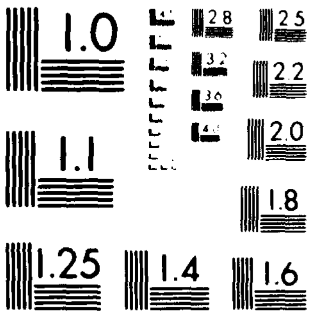
AD-A084 998 ILLINOIS UNIV AT URBANA-CHAMPAIGN COORDINATED SCIENCE LAB F/G 5/8
VISUAL RECOGNITION OF ARTIFACTS BY COMPUTER.(U)
AUG 79 C J JACOBUS DAAB07-72-C-0259
UNCLASSIFIED R-851 NL

1 of 2
AD
A084998




1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A084 998	3. RECIPIENT'S CATALOG NUMBER (9) Doctoral thesis
4. TITLE (*and Subtitle*) (6) VISUAL RECOGNITION OF ARTIFACTS BY COMPUTER		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER (14) R-851, UILU-ENG-78-2244
7. AUTHOR(s) (10) Charles Jeremiah Jacobus		8. CONTRACT OR GRANT NUMBER(s) (15) DAAB-07-72-C-0259, DAAG-29-78-C-0016, N00014-79-C-0424
9. PERFORMING ORGANIZATION NAME AND ADDRESS Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (12) 184
11. CONTROLLING OFFICE NAME AND ADDRESS Joint Services Electronics Program		12. REPORT DATE (11) Aug [redacted] 79
		13. NUMBER OF PAGES 175
14. MONITORING AGENCY NAME & ADDRESS (*if different from Controlling Office*)		15. SECURITY CLASS. (*of this report*) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (*of this Report*)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (*of the abstract entered in Block 20, if different from Report*)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (*Continue on reverse side if necessary and identify by block number*)

Graphs
Graph Matching
Edge Detection
Region Analysis
Scene Analysis Object Modeling

20. ABSTRACT (*Continue on reverse side if necessary and identify by block number*)

This work describes a set of programs which implements a bottom-up artifact modeling and recognizing system. This system is implemented on a DEC KI-10 in BLISS-10, MACRO-10, and FORTRAN. Sequences of steroscopic pairs of images are digitized, frame by frame condensed to vertex-string-surface graphs (V-S-S graphs), and re-encoded as half chunk graphs (H-C graphs). The single frame processing requires approximately 20 minutes per image. Single image derived H-C graphs are matched by stereo pairs for depth, and by time interval pairs for motion. By using depth cues, motion

DD FORM 1 JAN 73 1473

UNCLASSIFIED [illegible]

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

20. ABSTRACT (continued)

cues and intensity feature labels, individual object subgraphs are segmented. Individual object graphs are matched with and/or entered into an object graph library.

The report ~~We~~ describes two new edge detection algorithms, an edge-based region aggregation algorithm, a scan line oriented vertex-string encoding algorithm, a half chunk graph matching algorithm, and a histogram-based graph matching algorithm. ~~We also introduce~~ the idea of the "half chunk", an elemental curvature element which can be used to form scale and coordinate system invariant object graphs (or "feature" centered object models) is introduced

UILU-ENG 78-2244

VISUAL RECOGNITION OF ARTIFACTS BY COMPUTER

by

Charles Jeremiah Jacobus

This work was supported in part by the Joint Services Electronics Program (U.S. Army, U.S. Navy and U.S. Air Force) under Contracts DAAB-07-72-C-0259, DAAG-29-78-C-0016 and N00014-79-C-0424.

VISUAL RECOGNITION OF ARTIFACTS BY COMPUTER

BY

CHARLES JEREMIAH JACOBUS

B.S., University of Illinois, 1973
M.S., University of Illinois, 1975

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1979

Thesis Adviser: Professor R. T. Chien

Urbana, Illinois

VISUAL RECOGNITION OF ARTIFACTS BY COMPUTER

Charles Jeremiah Jacobus, Ph.D.
Coordinated Science Laboratory and Department of Electrical Engineering
University of Illinois at Urbana-Champaign, 1979

This work describes a set of programs which implements a bottom-up artifact modeling and recognizing system. This system is implemented on a DEC KI-10 in BLISS-10, MACRO-10, and FORTRAN. Sequences of stereoscopic pairs of images are digitized, frame by frame condensed to vertex-string-surface graphs (V-S-S graphs), and re-encoded as half chunk graphs (H-C graphs). The single frame processing requires approximately 20 minutes per image. Single image derived H-C graphs are matched by stereo pairs for depth, and by time interval pairs for motion. By using depth cues, motion cues and intensity feature labels, individual object subgraphs are segmented. Individual object graphs are matched with and/or entered into an object graph library.

We describe two new edge detection algorithms, an edge-based region aggregation algorithm, a scan line oriented vertex-string encoding algorithm, a half chunk graph matching algorithm, and a histogram-based graph matching algorithm. We also introduce the idea of the "half chunk", an elemental curvature element which can be used to form scale and coordinate system invariant object graphs (or "feature" centered object models).

ACKNOWLEDGMENT

I would like to thank my advisor Prof. R. T. Chien for providing support, his interest and time during my time at the Coordinated Science Laboratory.

I would like to thank J. Michael Selander for the use of some of his software (a region growing system and his PDP-11 based graphics), and for the fruitful discussions we have had about the nature of vision.

I would also like to thank Dr. David Waltz, for his interest in my work, Rod Fletcher, for our discussions about curve fitting and vision hardware, Dr. Vincent Jones and Dr. Wesley Snyder, for the image software I inherited from them, John Paul and Tom Burtnett, for giving me free access to the computing facilities, Mrs. se Schmidt and Mrs. Barbara Champagne, for helping with administrative details and reminding me of all the university deadlines I would otherwise have missed, and Jack Gladin, for excellent photography on short notice.

I am grateful to my family for having put up with a perpetual student for all this time and am happy to finally be able to tell them that I am done.

Finally, I would like to thank Dr. Clifford Geschke and his wife, Sue, for making my first years in graduate school bearable, and my love, Heidi, for making the last wonderful.

TABLE OF CONTENTS

Page

Chapter 1

Motivations and Strategies

1.1 Motivations

This thesis consists of discussions of several operational programs which together embody an artifact recognition system. According to the _American Heritage Dictionary of the English Language_, an artifact is:

"An object produced or shaped by human workmanship; especially, a simple tool, weapon, or ornament of archaeological or historical interest."

We use the word artifact to mean a constructed object. As such, we design it to take a specific shape. Our visual recognition system is therefore designed to extract and manipulate shape related models of objects. These models are "discontinuity"-based. We encode objects as a cloud of discontinuity points (each carrying property lists which allow point to point comparisons) which are interconnected into graphs. The links between discontinuities represent relationships into which the connected points enter.

Recognition of a thing or a relationship, in the case of machine intelligence, can be defined as "instantiating" that thing or relationship. By this we mean producing a machine state which represents the thing or relationship. In our system, this state comes about by detecting critical primitive components (edges, regions, contours, etc.), encoding them as atomic quantities (building primitive data blocks in memory with feature vectors describing specific component variations), and forming relationships between these atoms (building graph structures which represent objects and object groupings).

In this work we have proposed and implemented a variety of specific algorithms for specific recognition and detection problems, however the primary contribution made has been the integration of many different processing steps into a bottom-up, model-based vision system. We began our design assuming that top-down processing was desirable for actual object cognition. To perform a model directed parsing, we decided it would be necessary to get initial correspondences from models to scenes via some bottom-up process. This bottom-up process, as it became more and more powerful, eventually became the entire recognition system. While we do not totally debunk top-down systems, we simply point out that carefully constructed bottom-up processing, with allowances for multiple interpretations, can perform recognition (as sensors become better, computers become faster, and algorithms more powerful). Whereas, top-down algorithms always require bottom-up starter processes.

1.2 Strategies

Our early visual processing is single image based. We have found that processing on the actual image arrays is the most efficient way to perform local convolution operations. But, because any operation performed uniformly over an entire retinal field (in our case nominally 256x256) is very slow for serial computers (it takes approximately 8 seconds for our DEC KI-10 to sweep through an image array for averaging operations), we wish to move into more abstract spaces quickly (into spaces described by image graphs).

We begin by performing edge detection (Chapter 2). We have invented two new edge detectors which allow detection of sharp and diffused intensity transitions. We have also evaluated these two detector schemes along with some other techniques described in the literature. Our techniques are good for detecting shadow and object boundaries in a noisy environment, if the noise statistics are known and sufficiently below the level of significant intensity changes. These restrictions apply readily for vidicon and solid state visual band mosaic sensors.

Following edge detection, we segment regions by convolving homogeneity operators with the edge detected images. We use the absence of edges in an area to form connected regions. In effect, we fill in broken edge chains which enclose compact areas, and generate some classes of subjective contours (Chapter 3). We then use a scan line based (non-following) algorithm to find vertex points and build contour strings. These strings are smoothed and written as vertex-string-surface graphs (V-S-S). The contour encoding is made simpler by the presence of accurate region assignments prior to boundary extraction.

Color images can be processed by performing three times as much work prior to region and vertex-string processing (by edge detecting on red, green, and blue images - superimposing the results). For textured scenes s_nificantly more processing is required. Rough areas are transformed into smoothed areas in new images. These images are re-edge detected. The new edge data is precipitated into the old edge images from the original intensity image (after edges in rough regions have been removed). Therefore, the amount of additional processing for texture data is proportional to the number of individual texture smoothing processes implemented (for a good texture system probably 15 to 100 times more processing would be required).

Following V-S-S formation from single images, we no longer operate on the sampled image space. We transform the V-S-S graph into a half chunk graph (H-C). The half chunk graph encodes scenes as interconnected elemental curvature elements (the half chunks), each having a property list contain image intensity properties and spatial properties (positions, tangents, and tangent changes). For the domain of artifacts, these curvature elements contain the critical object features required for recognition, and allow object encodings resistant to changes in viewing angle. We correlate feature groupings between pairs of H-C graphs for depth and motion information (Chapter 4). The result of each graph matching is a transform which takes one graph dimensionally into the other, and a new fused graph which is made by merging as many compatible features as possible between the input graphs.

Following graph matching many object subgroupings have been disassociated (parts of single objects are connected in depth and nave common motion transforms, however parts of different objects tend to be disconnected or do not move together). To further disassociate object groupings a "Waltz-liks" labeling scheme is employed. This process is based on vertex and region labeling. Labels are constrained by semantic types and measured boundary feature properties (Appendix D).

Our object recognition program operates on H-C graphs which are assumed to be single objects (or descriptive fragments of single objects). The advantage of the H-C graph for object modeling is that it is "feature" centered rather than "object" centered. The graph is scale and coordinate system invariant (Chapter 5). We match object H-C graphs by extracting histograms of subgraph components between input graphs and graphs stored in an object library. These histograms are then compared. It is proposed that objects be modeled as the histograms alone. It is also proposed that the H-C graphs be directly generated from edge data (without the intermediate step of V-S-S graph formation).

Chapter 2

Two New Edge Detectors

2.1. Introduction

We have done extensive experimentation in low-level, bottom-up segmentation and have found edge detection based techniques to be very tractable from both theoretical and practical standpoints. We have in the process of this work designed two interesting detection methods, one difference-based and the other adaptive threshold based. These techniques are desirable because they find "thin" edge strings (i.e. unlike strict gradient-based approaches where thick bands of edge data are generated), they may be tuned to give good signal/noise ratios for a variety of edge transition sizes and image noise statistics, edge position may be computed to greater precision than the basic pixel size, and implementation is efficient in hardware or software. Our techniques are based on the application of arrays of edge detectors, each sensitive to a different group of edge types. An algorithm for selecting the most appropriate detector response in a particular region is used to consolidate the results for all detectors.

2.2. Related Work

Most of the work in low-level visual segmentation can be grouped into one of five main categories. These are segmentation by feature histogram, by region-based processing, by gradient-based edge detection, by edge detection with non-maxima suppression, and by edge template matching. Our work has elements of last two techniques.

The classic work in the area of visual segmentation by histogram based techniques was done by Ohlander [45]. In his work, one dimensional histograms of various color features were partitioned using a heuristic concept of histogram peak "goodness" (a combination of peak shape and isolation). Then the partitions were reprojected into image space to generated the required object partitions. Similar work has been done by Schacter, et al. [56] and Hanson [20], using two dimensional histograms, with similar results.

These histogram-based techniques allow good results for some cases, however they ignore some fairly important problems. First, it has been our experience that camera systems tend to have slow, uniform intensity variations over the retinal area which are not caused by lighting or object reflectance. In color imaging, this variation does not track well from color spectrum to color spectrum. What is worse, the amplitude of the variations may change with overall ambient lighting. In summary, all the points associated with a given surface may not give rise to a well defined peak in the histogram space. Obviously, histogram peak selection can then break down.

A tacit assumption of histogram-based techniques is that there is a one-to-one relationship between histogram peaks and object surfaces. Under this assumption, each peak may be thought of as an independent, unimodal feature distribution function, and peak selection is simply the identification of the parameters of the distribution. The basic problem in histogram techniques can be traced back to this assumption. While in many cases each peak is predominantly from a single object surface, we have absolutely no guarantee that this is the case. When it is not, histogram techniques are not valid.

Region-based techniques have been used by Brice and Fennema [6], Tenenbaum [60], Feldman and Yakimovsky [14], and others. The basic approach consists of selecting a region seed point (possibly any point, or alternatively some archetypical point), and adding adjacent unassigned points minimizing some difference criterion. Region-based techniques are theoretically equivalent to edge base techniques except to the extent that they incorporate higher level information. Without this high level information the contours generated between regions will exactly coincide to contours of maximum difference (the same as the contours generated by a non-maxima suppressing edge detection scheme). Brice and Fennema incorporated a figure "goodness" criterion by allowing region boundaries to grow at a constrained rate. Tenenbaum uses a semantic criterion in addition to simple intensity information to order merging. Feldman and Yakimovsky incorporate a Baysian decision technique into their region-based system.

We feel that the initial feature extraction phase of processing is not the place to incorporate high level knowledge. If this is done, it becomes difficult to evaluate performance and detracts from overall low-level system generality. The system starts seeing what it wants to see, rather than what is actually there.

We have expanded edge detection based attacks on the segmentation problem into three different approaches because each represents a slightly different view of what an "edge" is. In the pure gradient approach, an edge is a high gradient point. This view of things goes back at least to Roberts [51], and has persisted in work by O'Gorman and Clowes [46], Shirai [58], and others. The problem with this idea of an edge is that over a thick band at surface-surface interfaces we get high gradient points.

Rosenfeld [52][53] and Burr [7] consider an edge point to be where the maximum (minimum) gradient occurs. This maximum point suppresses all non-maxima in some area of influence around itself. Rosenfeld, as an additional refinement, sweeps the image with operators of ascending size (1x1, 2x2, 4x4, 8x8, etc.) to average over random textures. His multiple operator scheme supresses non-maxima over each set of operators of a given size. Then over each area, the largest operator giving a strong gradient is selected, suppressing smaller operators in the same area. In Rosenfeld's paper it can be seen that in textured environments this approach has the desired effect. However, in the domain of machine parts, where objects themselves display fine detail, Rosenfeld's algorithm for multiple edge detection does not work. It is to this problem we address ourselves.

An alternative definition of an "edge" is any intensity pattern which fits an "edge" template. The discrete difference mask used by Roberts could be viewed as a trivial template. Selection of a maximum point could be viewed as selecting a point where the difference template correlates with the image maximally. Several operators of a more complex nature come about by viewing edges in this light.

The operators of Yakimovsky assume edges are interfaces between sets of points, each set being described by a normal distribution. The mathematics for distribution parameter comparison is used to form a function of edge strength in an area.

2.2.1.
$$S < \frac{(\sigma_0^2)^{m+n}}{(\sigma_1^2)^m (\sigma_2^2)^n}, \quad \text{for } N(\mu_1, \sigma_1) \neq N(\mu_2, \sigma_2)$$

where:

$\sigma_0^2 =$ variance for both neighborhoods taken together

$= [m\sigma_1^2 + n\sigma_2^2 + m(\mu_0 - \mu_1)^2 + n(\mu_0 - \mu_2)^2]/(m+n)$

$\mu_0 =$ mean for both neighborhoods taken together

$= (m\mu_1 + n\mu_2)/(m+n)$

$m, \mu_1, \sigma_1^2 =$ samples, mean, variance for neighborhood 1

$n, \mu_2, \sigma_2^2 =$ samples, mean, variance for neighborhood 2

We have, for comparison purposes, programmed an operator of this sort with non-maxima suppression, using the neighborhood shapes proposed in [66]. It should be pointed out that the operator of Yakimovsky is in fact a superposition of the standard expressions for comparing means [30],

$$2.2.2. \qquad S < \frac{\sigma_0^2 (m+n)}{m\sigma_1^2 + n\sigma_2^2}, \text{ for } \mu_1 \neq \mu_2$$

and comparing standard deviations,

$$2.2.3. \qquad S > \frac{(m\sigma_1^2)^m \, (n\sigma_2^2)^n}{(m\sigma_1^2 + n\sigma_2^2)^{m+n}}, \text{ for } \sigma_1 \neq \sigma_2$$

This points up some problems. The mean comparison component behaves well when the operator is not centered on an edge interface (the mean comparison is more or less a finite difference, therefore an approximation to the gradient). However, the sigma component peaks at interfaces where standard deviation changes and on both sides of ones where the mean changes. This can cause false edges to be detected. The statistical assumption of two independent distributions is valid only right on surface-surface interfaces.

The operator of Hueckel [22][23][24] comes about by expressing a family of templates in terms of an orthonormal family of functions. We have programmed a version of the original operator, but have been thoroughly disappointed with the results. From the paper of Mero [43] it can be seen that the Hueckel type approach can be done with different basis functions quite efficiently. We have reason to believe that the original basis functions of Hueckel are less than satisfactory and the ones proposed by Mero are more advantageous. Using these basis functions it was shown that the Hueckel type operator is more or less equivalent to a gradient. It should be pointed out that like the gradient, on gradual intensity slopes thick edges (more than a single point wide) occur. Also, angular resolution for this type of detector turns out to less than one might expect.

The point to be drawn from the previous discussions is that most of the techniques proposed do not differ substantially from a theoretical point of view. Practically, performance is more a function of operator geometry and image noise statistics than the models selected for edge strength computations (all model thus far discussed eventually reduce to subtraction of neighborhoods). Our system uses a variable geometry to average over image noise.

2.3. Difference-Based Multiple Operator Technique

This edge detection system is based on the simple step edge detector. Detection of steps, that is, places where intensity distributions each having different means meet, can be accomplished optimally by simple differencing. The basic difference is then scaled by a factor involving the standard deviations of the two distributions or heuristically, a term based on the variation around the mean change (see formula 2.2.2). The formula may be slightly modified as follows if we assume equal neighborhood sizes and take $\sigma_1 = \sigma_2 = \sigma$ (the standard deviation for the overall image).

2.3.1. $$S = 1 + \frac{R^2}{2n-2} \;,\; R < \frac{|\mu_1 - \mu_2|}{\sqrt{\sigma^2/(n-1)}} \approx \frac{|\mu_1 - \mu_2|}{\sqrt{\sigma^2/n}}$$

2.3.2. $$R\sqrt{n\sigma^2} < |n\mu_1 - n\mu_2|$$

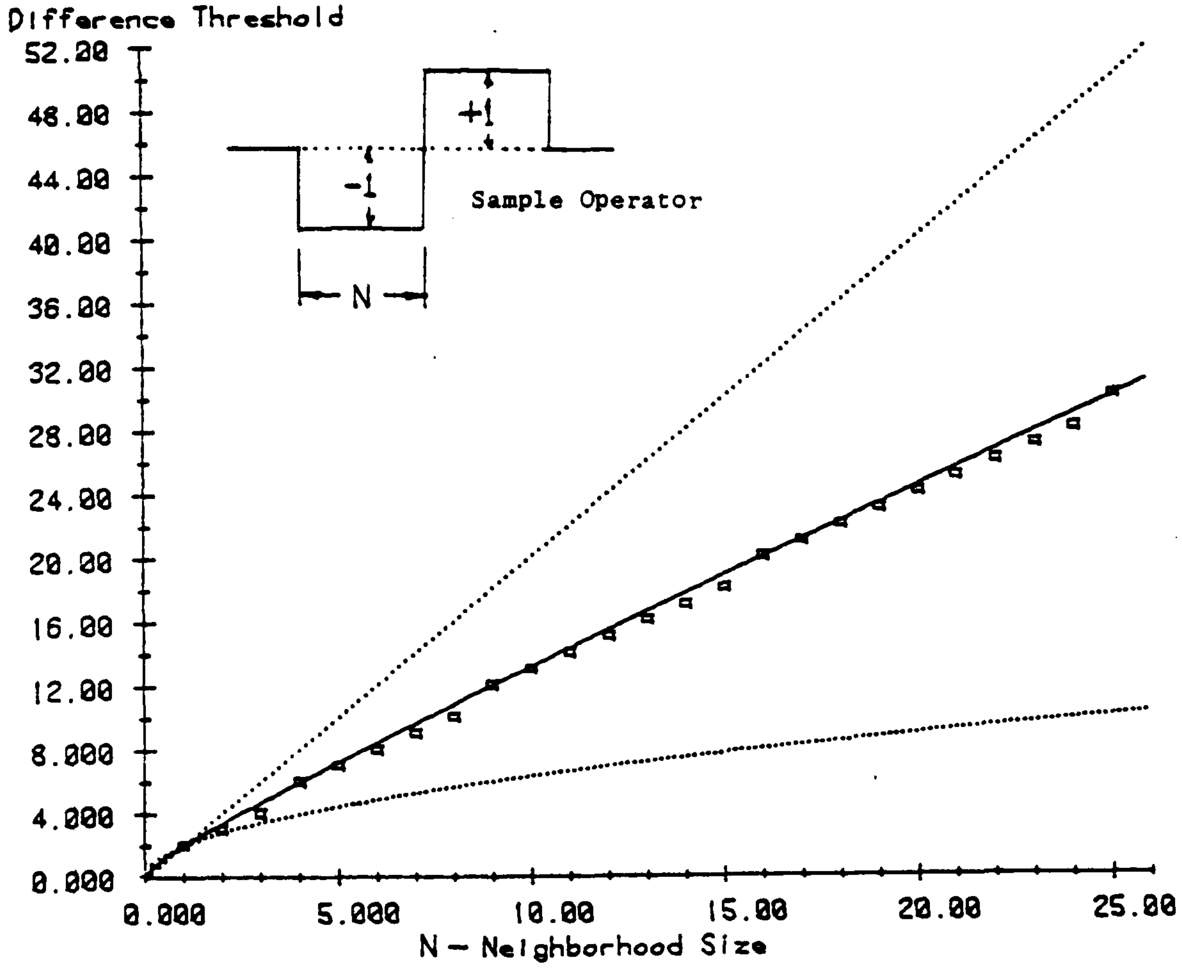
As noted in section 2.2 using the moments σ_1 and σ_2 to characterize texture is dubious because these terms respond very strongly also to mean variations, therefore we feel it reasonable to simply do the above scaling. If we assume t as the threshold for neighborhoods consisting of a single measurement then for neighborhoods consisting of n mesurements we expect the following:

2.3.3. $$T(n) = t\sqrt{n} \;,\; t = R\sqrt{\sigma^2} = R\sigma$$

2.3.3 is the result we would expect for purely Gaussian distributions, however our work indicates that the square root law actually generates thresholds that are too low. As a fix we can choose:

2.3.4. $$T(n) = t\left[(n - \sqrt{n})k + \sqrt{n}\right]$$

For our imaging system we have found thresholds plotted as boxes in Figure 2.1 are quite good (k=0.6). The constant k can be considered the degree to which difference due to "noise" accumulates linearly rather than normally.



Plot of T (Edge Detection Threshold) vs. N (Sample Size)
Figure 2.1

As is the case for any difference-based detector, there will typically be several adjacent points where edge strength is greater than T. We only want bands of width one. To accomplish this we observe that what goes up must come down, or the value of X in 2.3.1 alternates sign for each new step. A peak is recognized as all the points where X is greater than zero, or all the points where X is less than zero. The center of the peak is either its absolute maximum or its center of mass (generally either measure is relatively noise free and nearly located at the same position). The center of mass method is better in that it allows positioning more finely than the basic pixel size. Both methods yield edges of width one.

The use of relative maxima rather than absolute maxima thinning has been used extensively and is really at the heart Rosenfeld's notion of non-maxima suppression. The problems of using pure relative maxima techniques is clearly shown in Figure 2.4. In Figure 2.2 we show an intensity profile. Figure 2.3 shows peak selection using either the center of mass method (CM) or the absolute maximum method (MAX-MIN). Figure 2.4 shows multiple peaks detected where only one should be using relative maxima detection.

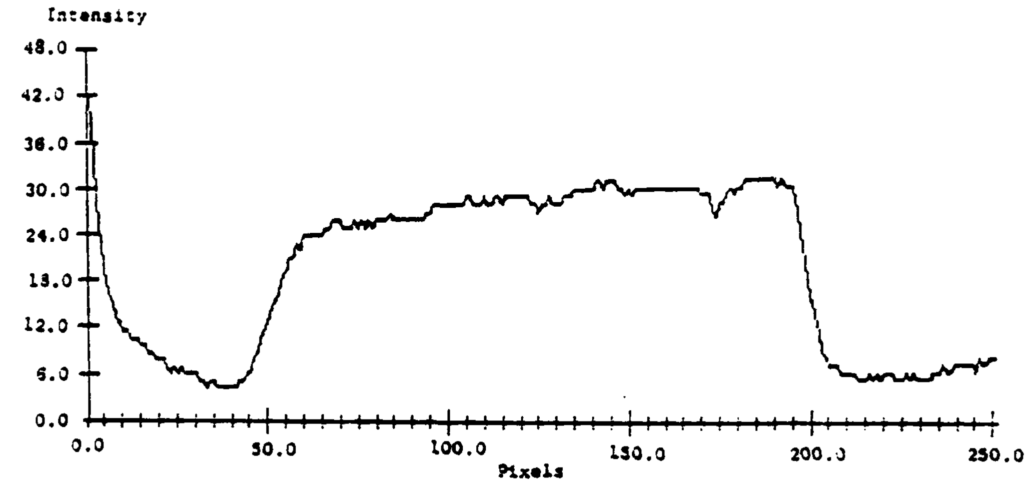


Figure 2.2. Intensity Profile

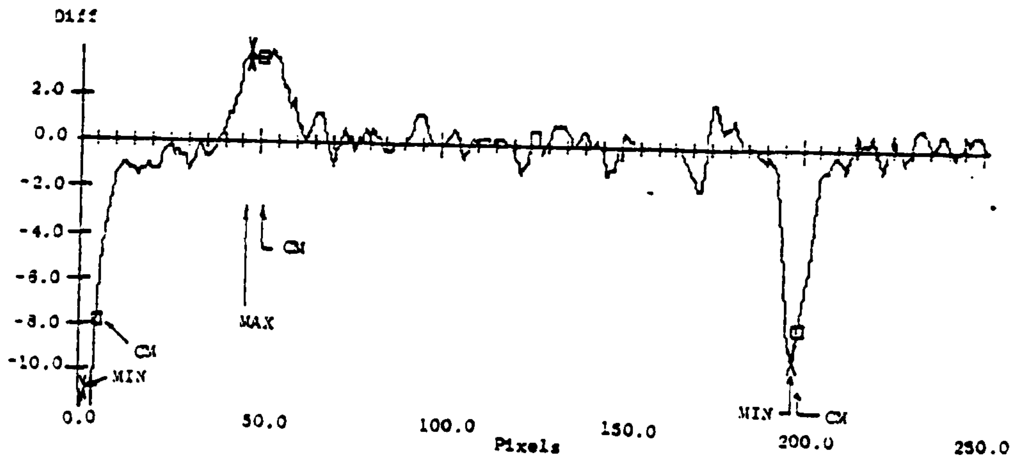


Figure 2.3. Difference With Center of Mass (Box) and Maximum (X) Peaks Marked

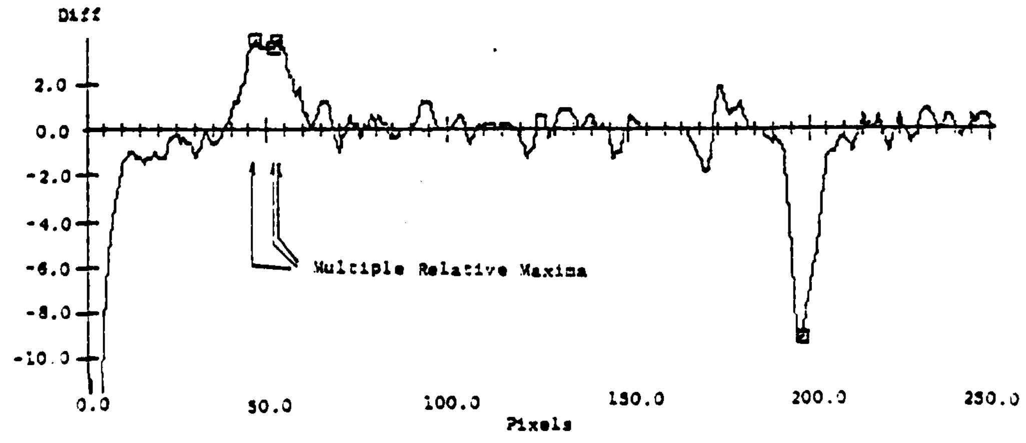
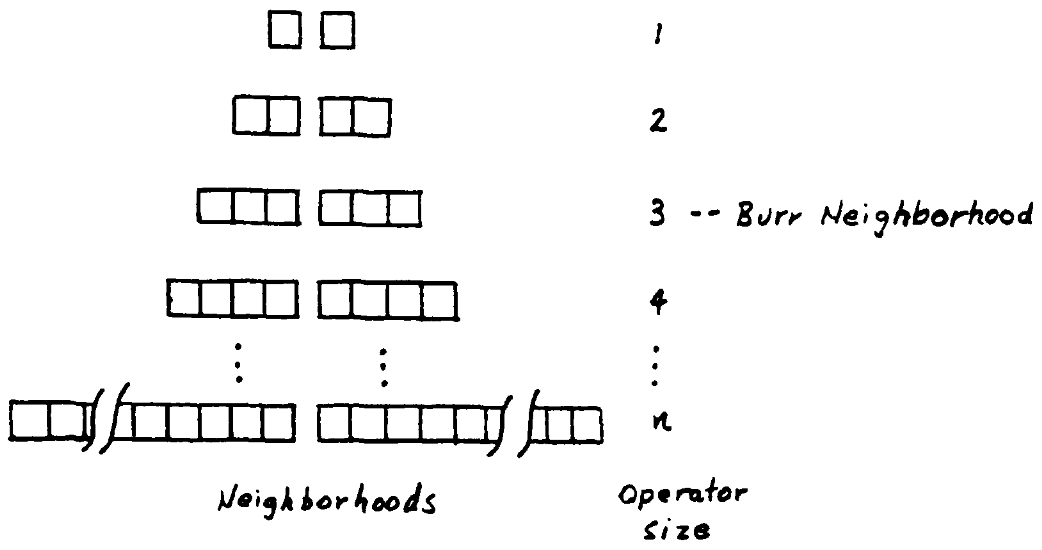


Figure 2.4. Difference With Relative Maximum Peaks Marked

In short, the single detector indicates one edge per peak for any peak having points with an edge strength greater that T. We have thus far restricted our discussion to one dimensional step detection. To expand to two dimensions, the one dimensional edge detectors are swept along each horizontal and vertical line, generating two basic edge directions (more orientations may be used in principle, however two is the minimum required).

The sweeping operation is performed for several sized operators at the same time, the larger differences being incrementally computed from the smaller ones (Figure 2.5). Larger sized differences allow more discrimination for diffused broad intensity transitions, such as those found in shadows cast by non-point sources and highlights. The fastest transition detector that has strength greater than its corresponding T marks the edge position. All detections from larger detectors are disabled to prevent feature smearing. Many low contrast contours can be discriminated only by detectors using many cells averaged (note Figure 2.1, the larger the operator sample the lower the effective threshold T per sample). Also the marking detector index number becomes the edge diffusion size (proportional to the edge peak width). Figure 2.6 shows a typical intensity profile. Figure 2.7 shows the output of a small detector. Figure 2.8 shows that a larger detector is able to pull more contours out of the background noise than the smaller one (Figure 2.7).



Operator Neighborhoods
Figure 2.5

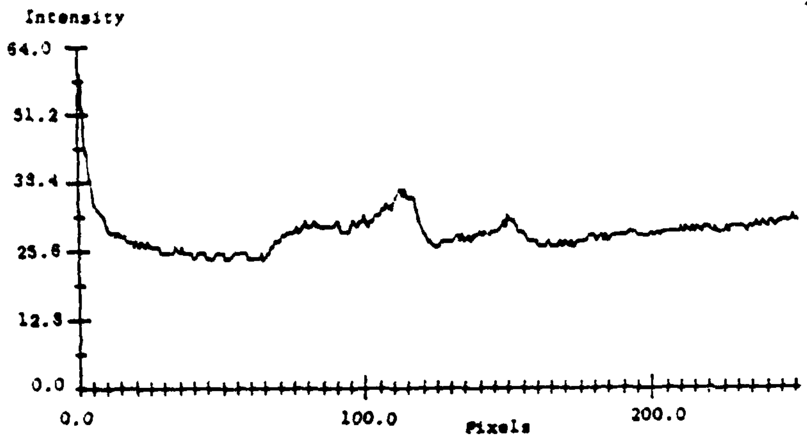


Figure 2.6. Intensity Profile

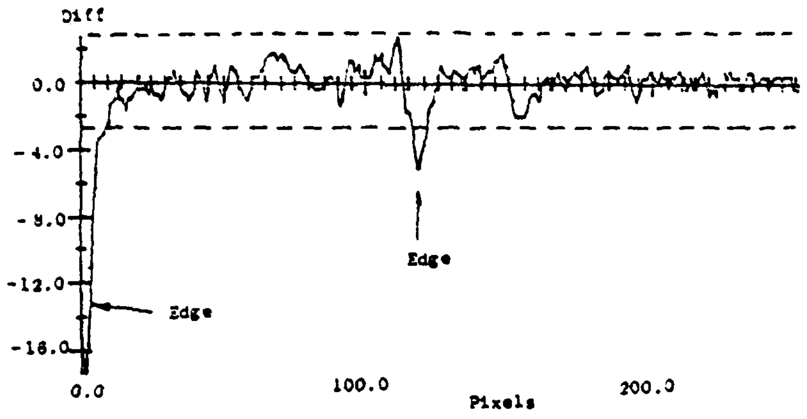


Figure 2.7. Difference, Size = 3

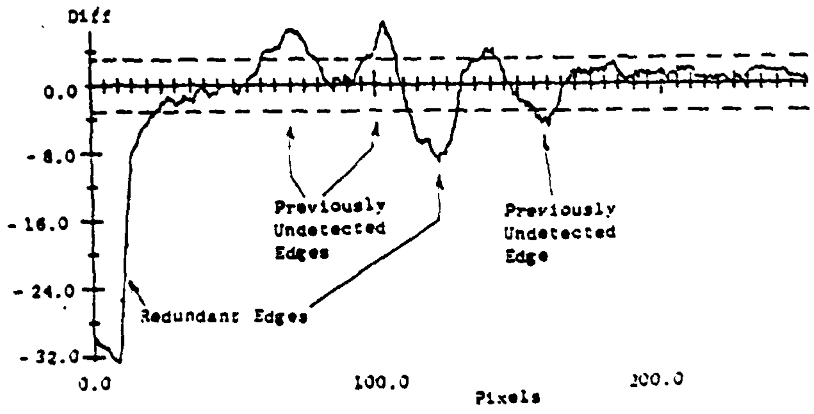
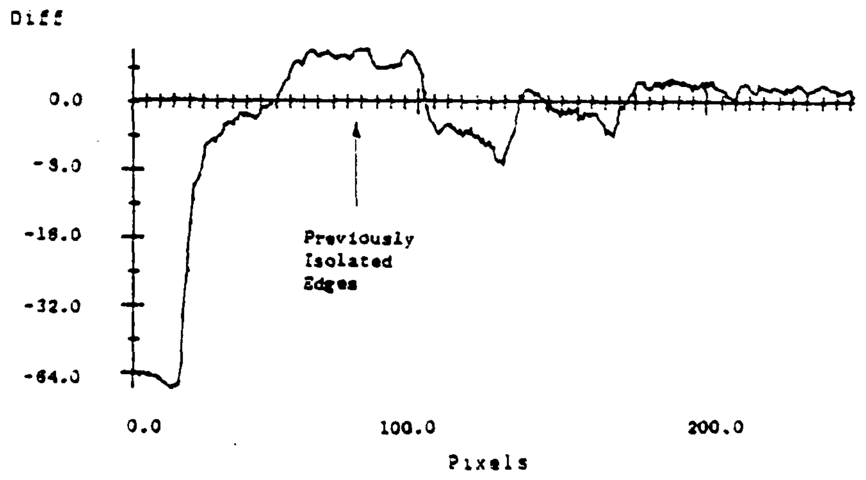


Figure 2.8. Difference, Size = 6

The preference towards small detectors is analogous to Marr's requirements of edge isolation, in the generation of his primal sketches [32][33][35] (really edge detected images). We wish the simplest explanation of intensity phenomena. To accomplish this we need to assume minimum interaction between edge detector outputs, i.e. isolated edge steps. The simplest way to minimize detector overlaps is to use the smallest detectors that give non-noise detections. If two small, isolated detectors fire (i.e. two successive edges), and over the same region, one larger detector fires, the simplest explanation is that the large detector is firing due to both small edges. Figure 2.9 shows how an operator can be large enough to smear together isolated edges.



An Operator Large Enough To Smear Edges
Difference, Size=12
Figure 2.9

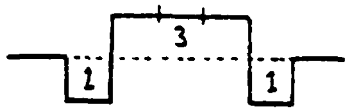
We may adjust the noise model (method of computing T for each detector) and the range of detectors (typically sizes 1 through 6) to fit a wide variety of imaging systems. It is our experience that this is a shortcoming of most other detection schemes. Contrasting this system and the system proposed by Rosenfeld [52][53] several major differences are apparent. In our scheme operator sizes are chosen in a linear sequence rather than a geometric one. The smallest operators have proportionally more influence than the larger ones, rather than the other way around as in the Rosenfeld scheme. In our system, low amplitude, random texture is ignored via an image noise model. We also characterize the size of the transition region for the intensity change.

2.4. Adaptive Threshold Multiple Operator Technique

After getting good results from the previous system of edge detection we wondered if we could get better results after some form of second order image enhancement. To our chagrin, the results were much worse. This was because the enhancement caused ringing at strong intensity steps. This ringing was picked up as "peaks" in the difference space and therefore caused multiple edges when of sufficient amplitude. These problems caused us to consider a non-differential approach to edge detection which is interesting for several reasons. First, the implementation is extremely fast and extremely simple. Peak selection and thinning are almost trivial. And last, the detection scheme has very nice biological analogs (works better with more enhancement, displays mach-band responses).

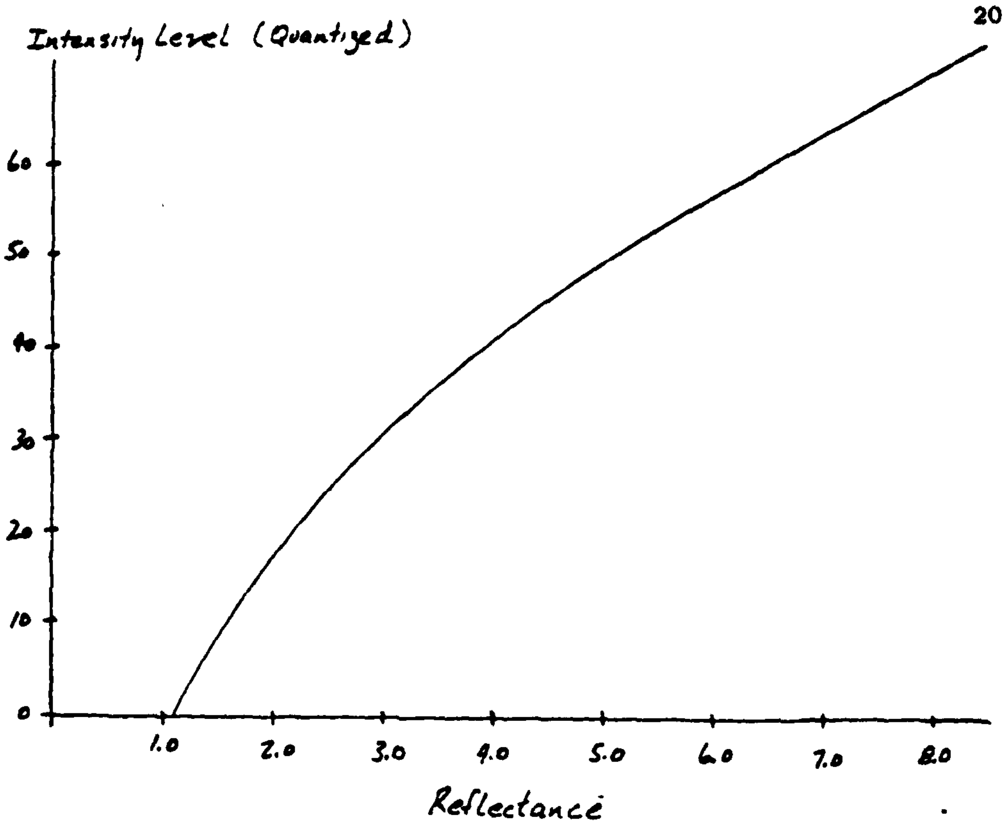
The first step is to compute from an input image a "fast" image and a "slow" image. The slow image is formed by computing the average intensity of a large neighborhood (in our case 20 by 20) centered at each cell. This picture is basically an artificially defocused image. It may be thought of as a "threshold" picture where each cell is proportional to the average light level in the larger neighborhood.

The fast picture is an edge enhanced picture. We do the enhancement in the X and Y directions independently using the function in Figure 2.10 evaluated at every cell position along a line orthogonal to the direction of the edges being extracted.



Enhancement Function
Figure 2.10

The slow picture is then subtracted cell-wise from the fast picture, generating a new picture which is zero in homogeneous regions and at edges, non-zero around edges. At this point we have computed a quantity analogous to the "lightness" proposed by Horn in [21], when we consider that the subtraction is actually comparable to a division because of our imaging system. This system has a log of light-level response (Figure 2.11 - this is generally true for vidicon imagers). This picture is then thresholded at plus or minus some threshold T. All cells with value V, $-T < V < T$ are set to 00. Cells with value V, $V > T$ are set to 01, and cells with value V, $V < -T$ are set to 10. In this way we obtain a three level picture (10,00,01).



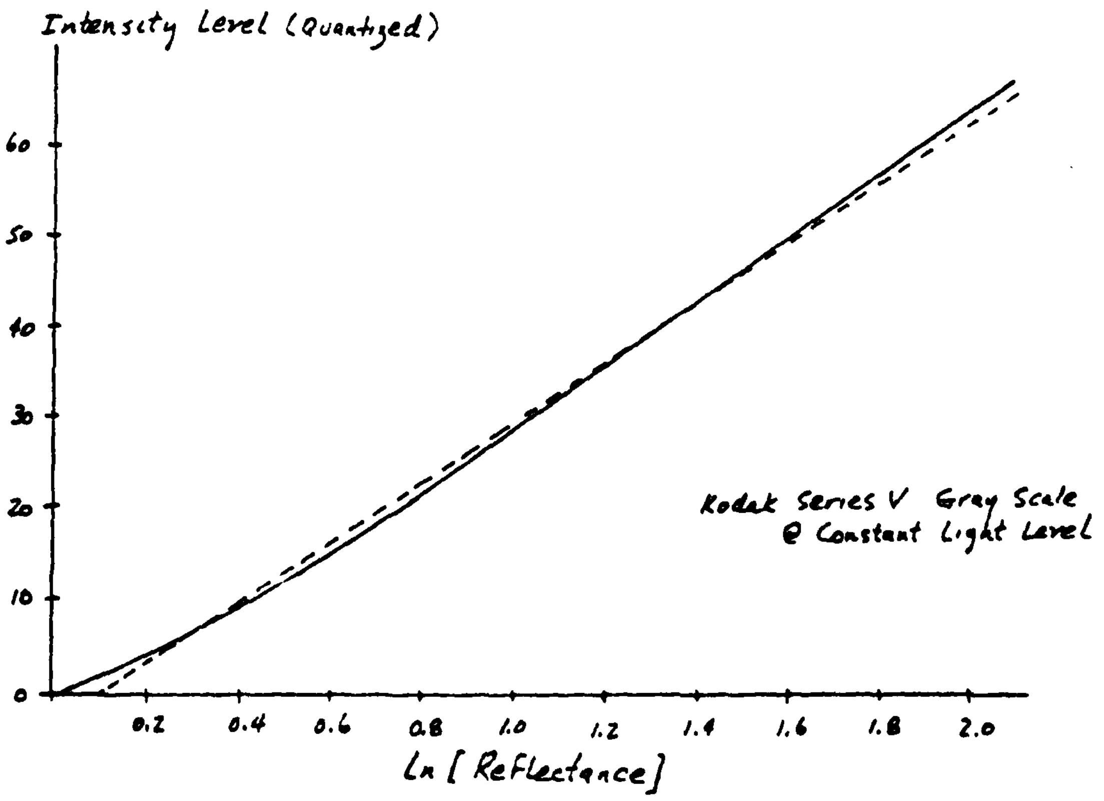


Figure 2.11. Quantization Level vs. Light Level and ln[Light Level]

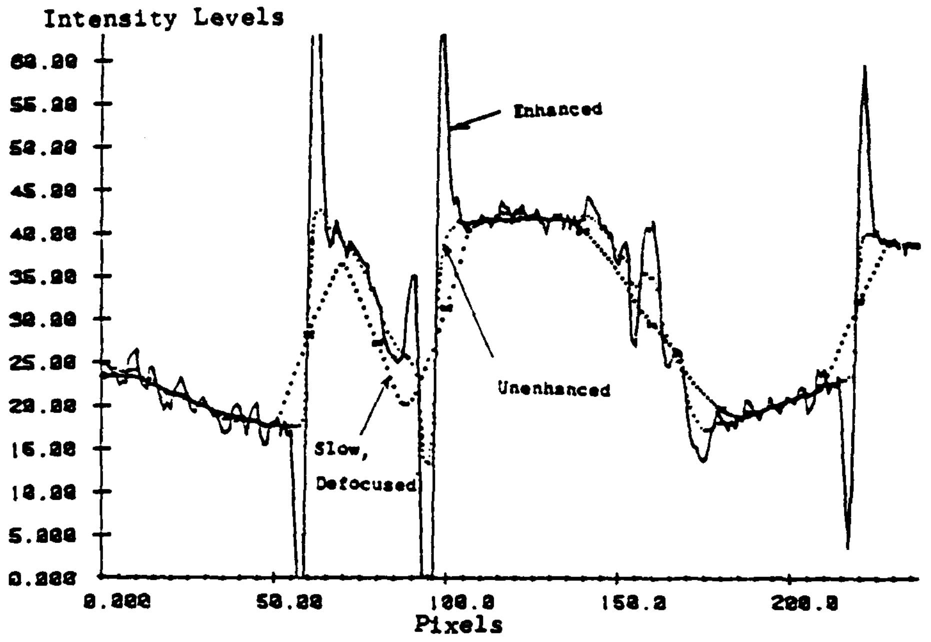
Edge extraction along a line now becomes a simple binary pattern matching process. An edge may now be expressed as:

2.4.1. <01><00 00 ... 00 00><10>

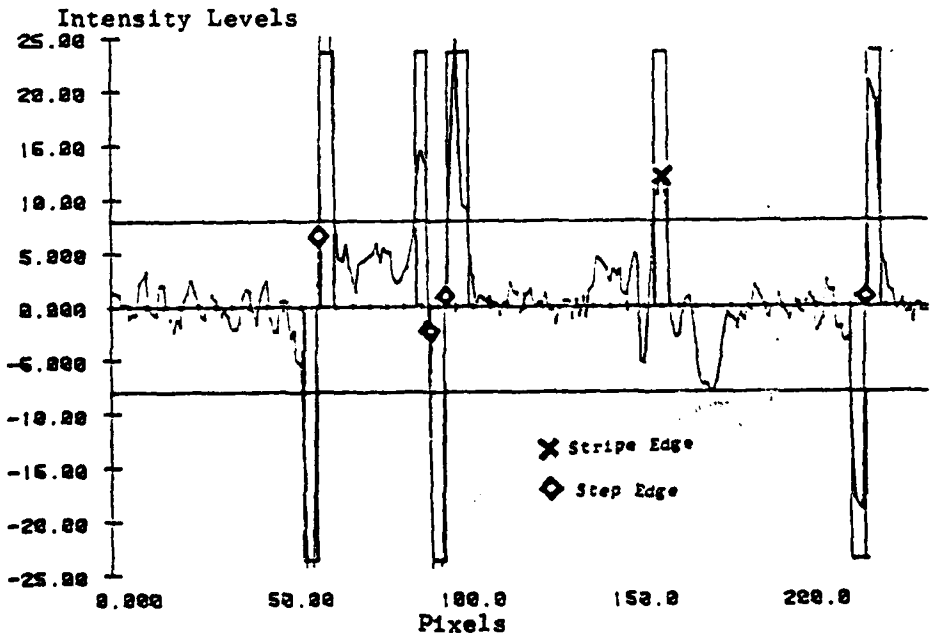
or

<10><00 00 ... 00 00><01>

The central string of 00 codes may have length zero for the fastest transition edges, or up to N where N is the longest acceptable transition. The leading 01 or 10 codes may be repeated. The first pattern corresponds to a negative going edge, the second, to a positive going edge. For each size edge template (sizes 1 to N), we sweep the three level pictures in the X and Y directions. We light a bit over each position where an edge template matches (there are N bit planes, one for each size template). Figure 2.12 shows an intensity profile, the defocused intensity profile, and the enhanced intensity profile. Figure 2.13 shows the subtracted profile, with peak areas marked(+0- three level data).

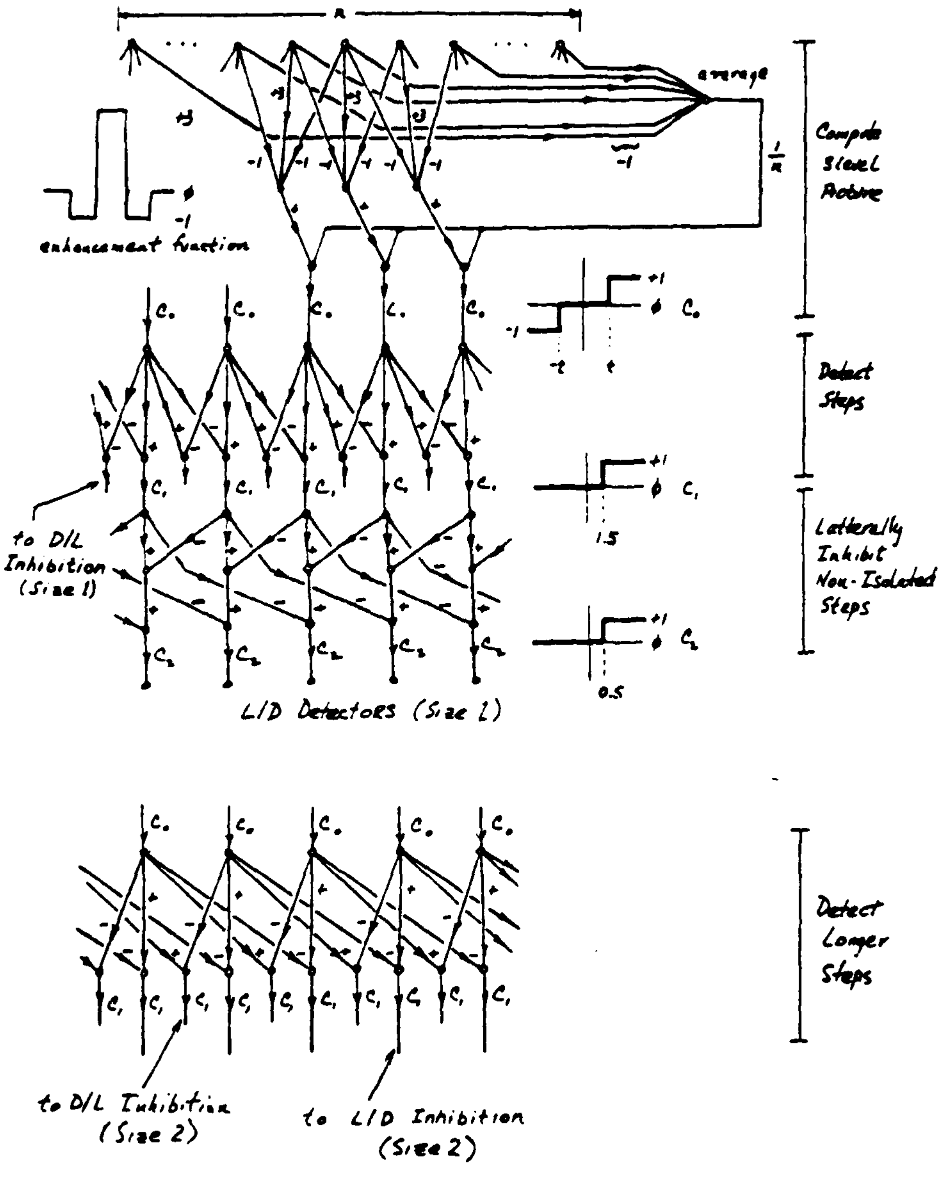


Fast and Slow Picture Profile
Figure 2.12



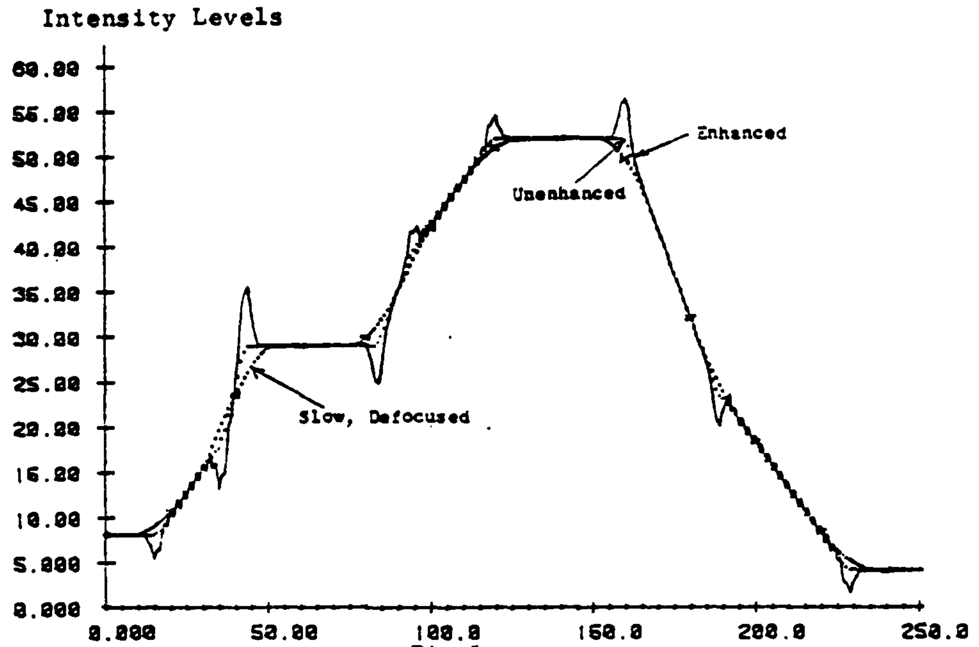
(Fast-Slow) and Three Level Data Profile
Figure 2.13

As indicated earlier, peak selection and thinning in the edge bit planes is trivial. Any cell in a given bit plane belonging to a string of edge points of length greater than two should be ignored. (There will be edges from edge templates with fewer 00 codes marking the step, having a run length of two or one. This is an isolation condition, analogous to small edge detectors preempting larger ones in our difference-based technique.) If a string of two is encountered, the position of the actual edge is between the two marks. If a singleton string alone is present, it directly marks the edge. We simply record mark position for edge position and bit plane index for transition region size. A hypothetical hardware implementation for one line of detectors is quite illuminating (Figure 2.14). We think the similarity in geometry between this structure and ones observed in biological visual systems is interesting.

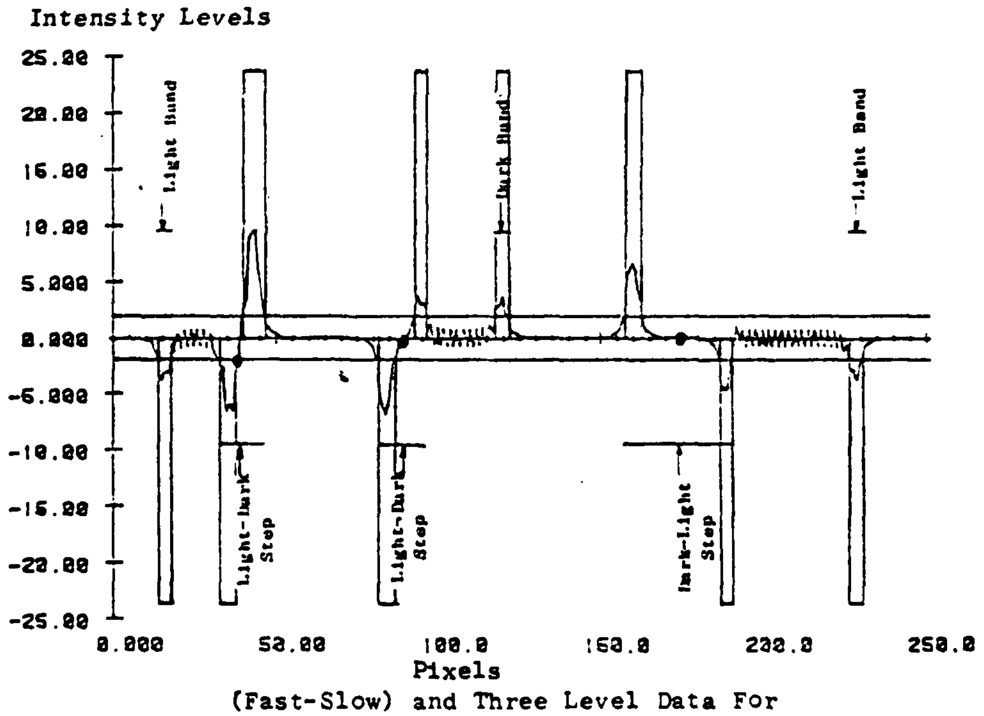


Circuit Diagram For One Line Of Detectors
Figure 2.14

Mach band response in this detector scheme comes about as a result of the second order enhancement used in generating the "fast" image. This enhancement causes the results in Figures 2.15-2.16, when the ramp breakpoint is sufficiently great to produce peaks of greater amplitude than T (threshold marked in Figure 2.15).

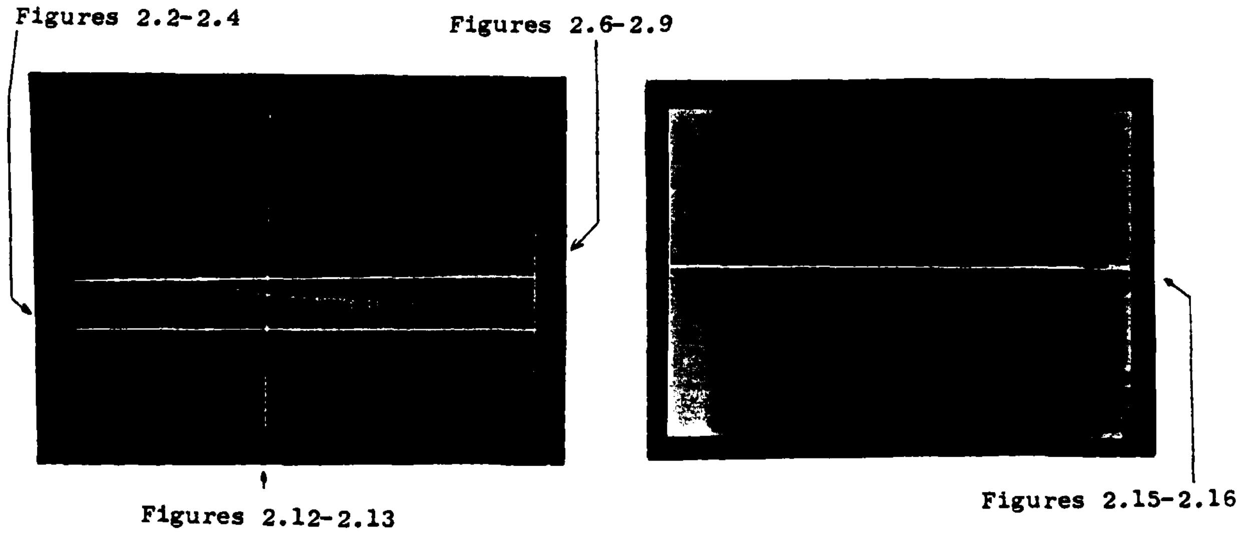


Intensity Ramp Profile
Figure 2.15



(Fast-Slow) and Three Level Data For
Profile of Figure 2.15
Figure 2.16

Figure 2.17 shows the images from which previously displayed intensity plots were derived. White lines mark the various slices.



Images Used For Intensity Plots
Figure 2.17

2.5. Performance Evaluations

To test the relative merits of our two edge detection schemes and those of several other researchers, we have prepared several characteristic test images. The first six are computer generated intensity ramps, one set of three vertically oriented (Figures 2.18-2.20), and the other set of three diagonally oriented (Figures 2.21-2.23). The first in each triple has no additive noise. The second has Gaussian noise of S=0.5 (nominally the noise induced in a quantization system - the noise levels encountered in our real images). The last in each set have Gaussian noise of S=1.0. These pictures test noise immunity, the degree to which edge detectors can follow diffusing steps, and to a limited extent, orientation bias.

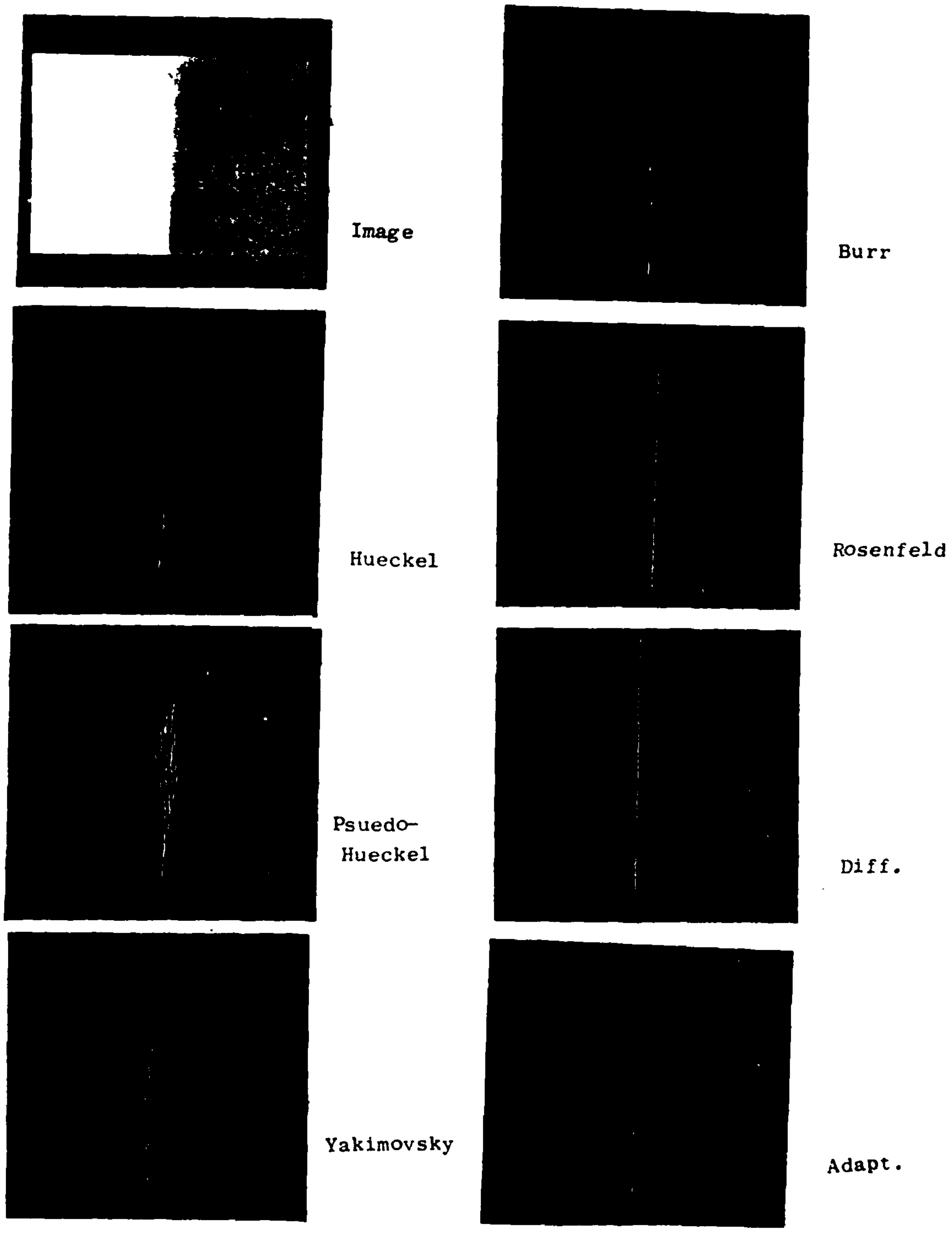


Figure 2.18. Vertical Ramp, S=0.0

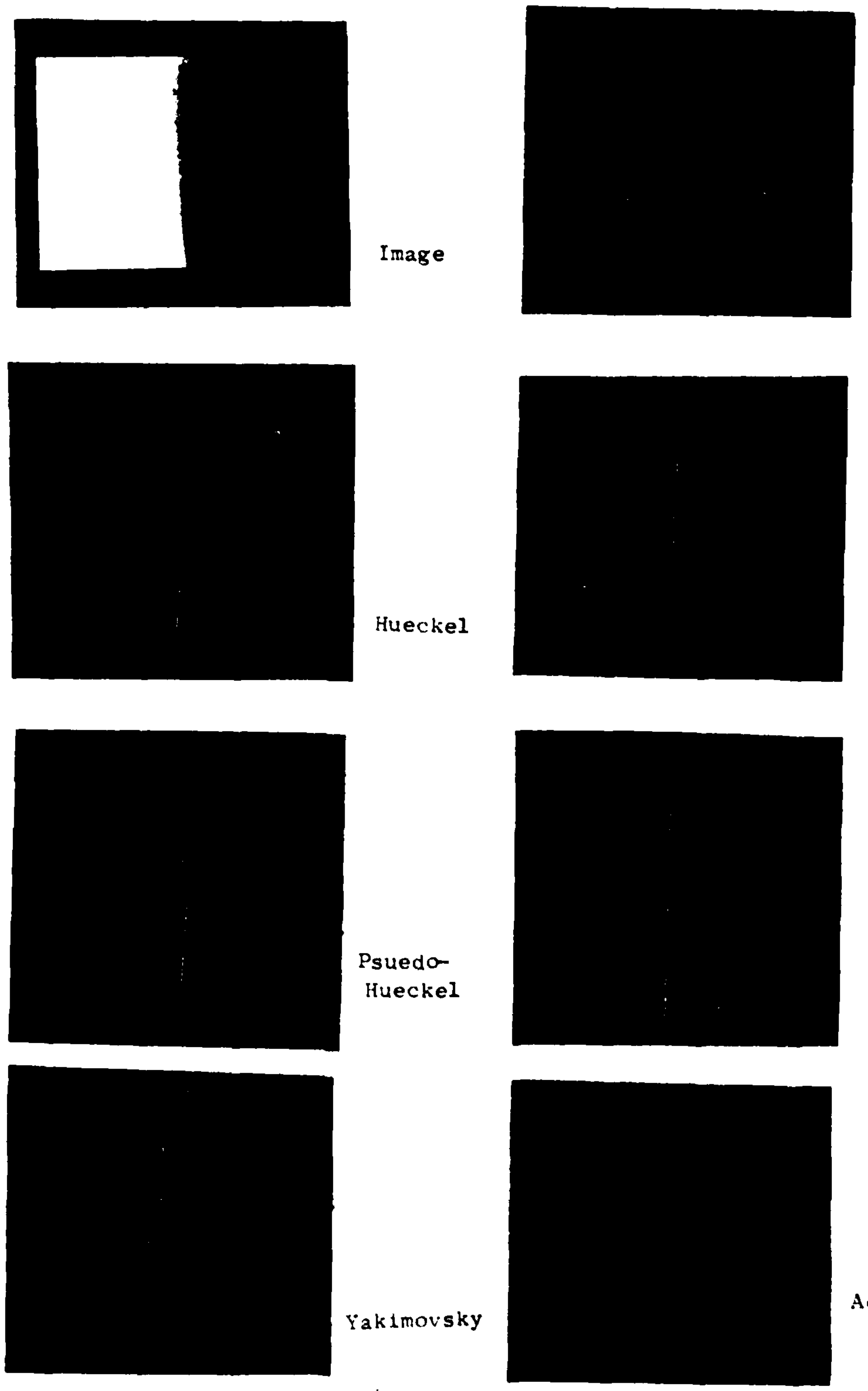


Figure 2.19. Vertical Ramp, S=0.5

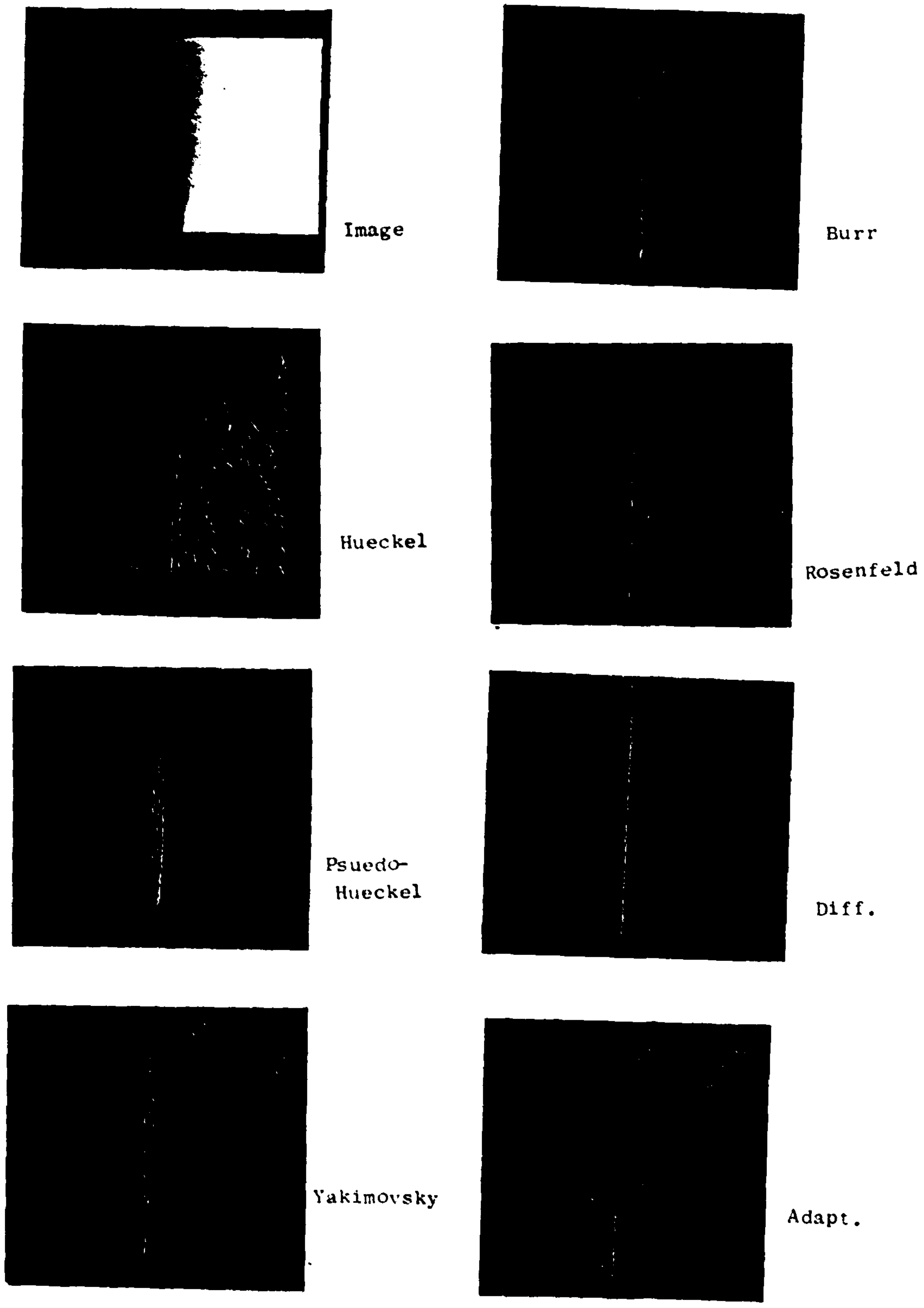


Figure 2.20. Vertical Ramp, S=1.0

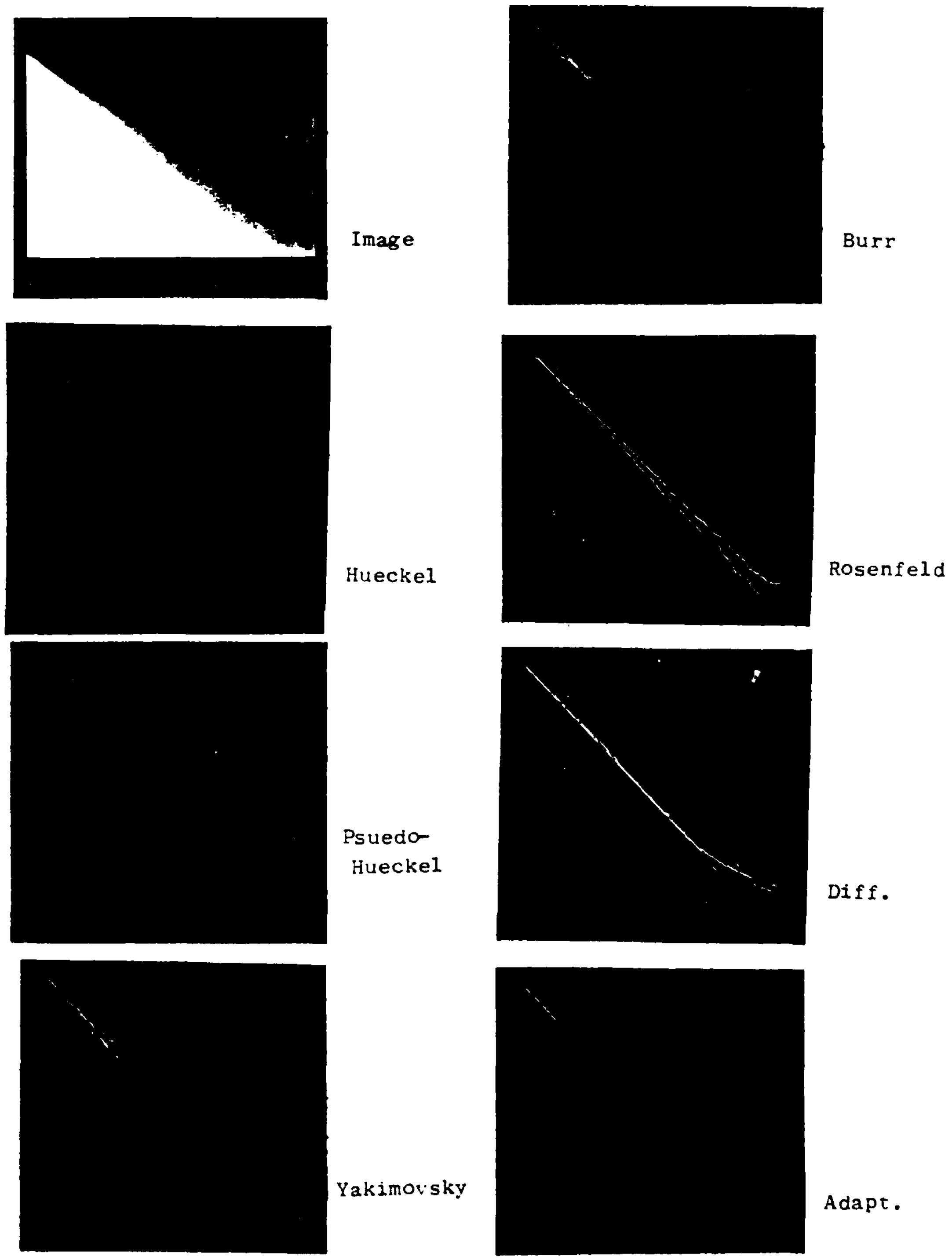


Figure 2.21. Diagonal Ramp, S=0.0

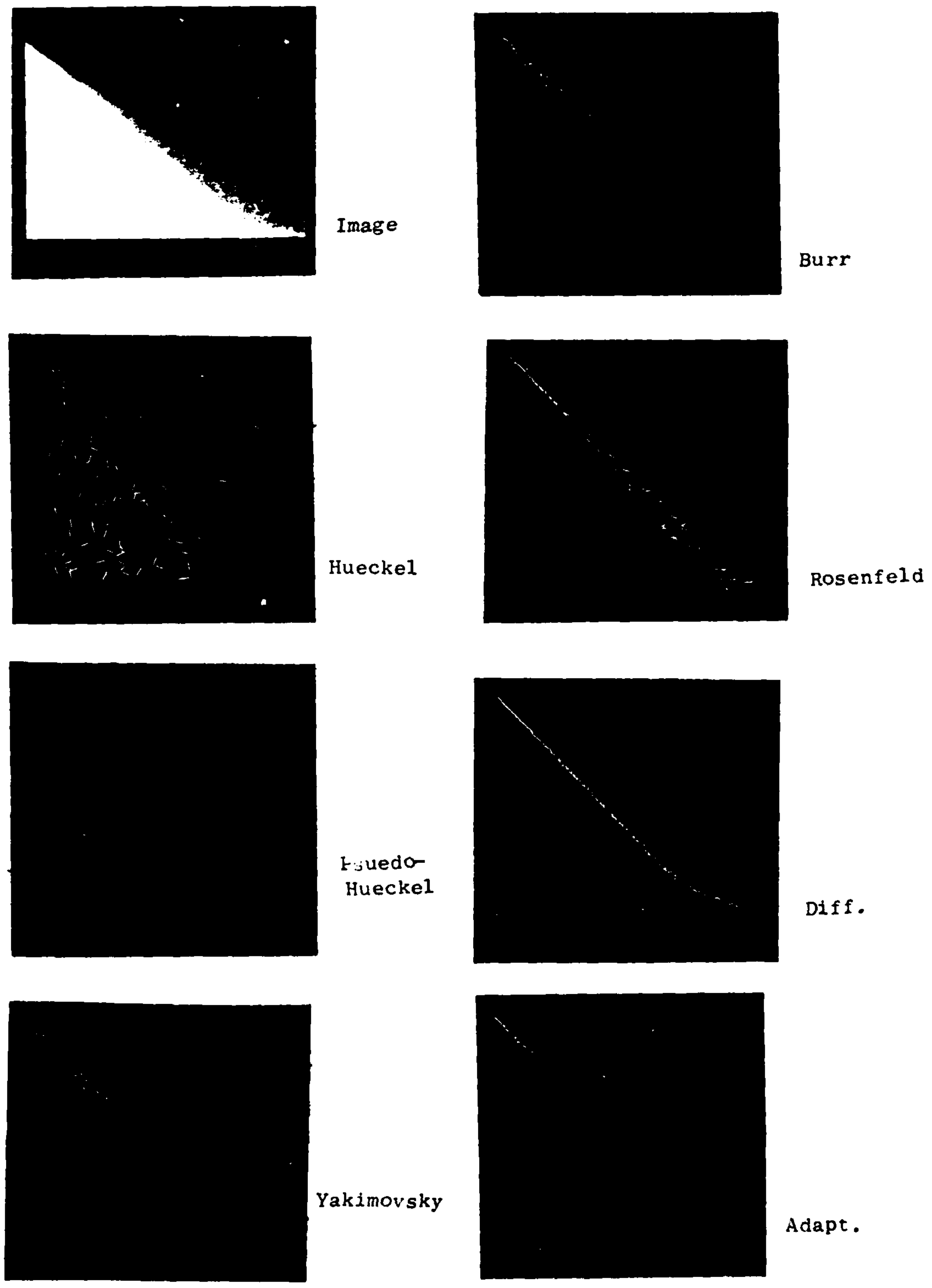


Figure 2.22. Diagonal Ramp, S=0.5

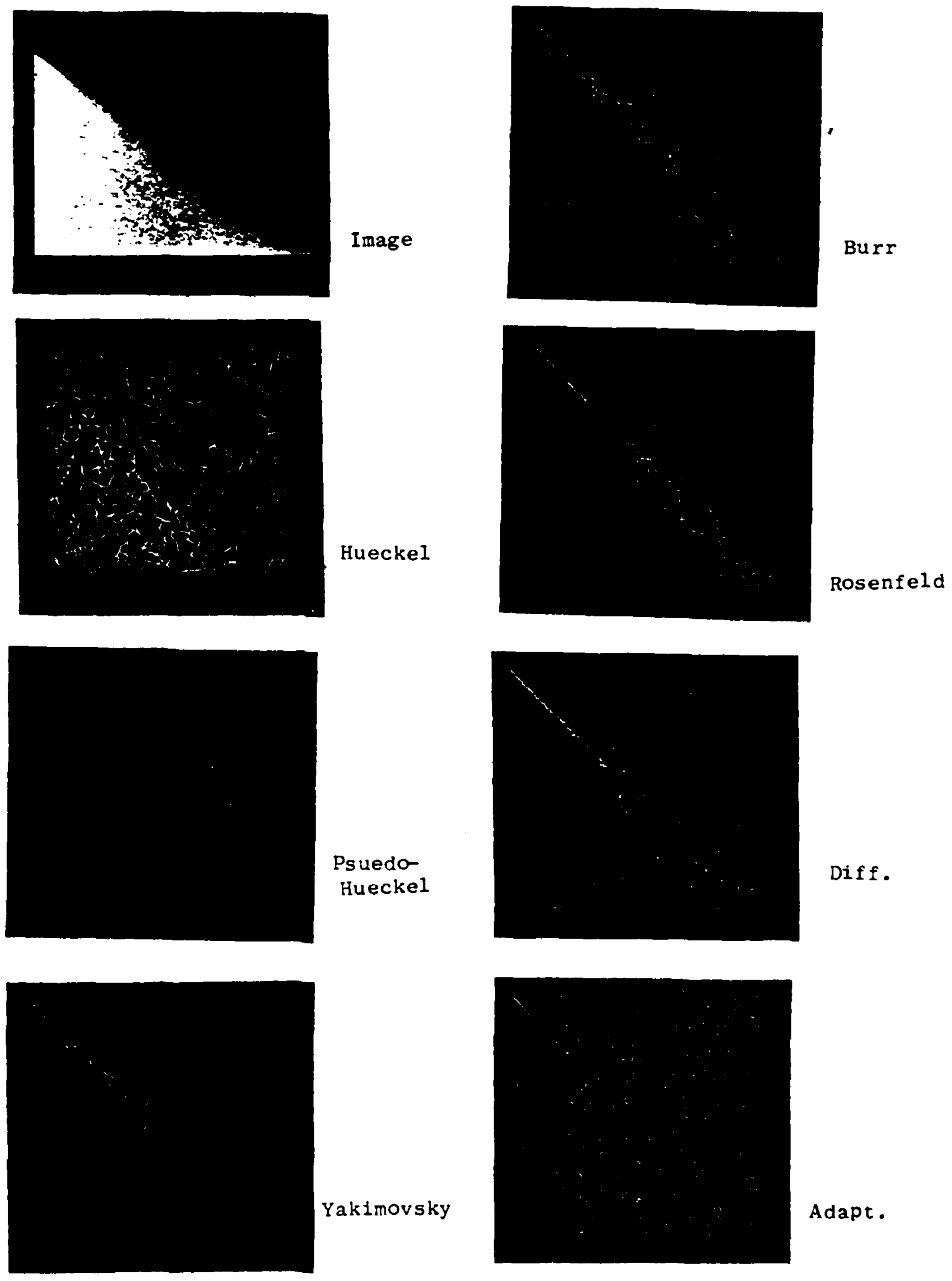


Figure 2.23. Diagonal Ramp, σ=1.0

We have also digitized some characteristic images taken in our laboratory to test the various edge detection techniques under more practical conditions. These images are 252x238x6 bits and have typically +1 or -1 level random intensity variation (occasionally +2 or -2). Figure 2.24 tests repetitive edge detections. Figure 2.25 tests the degree to which orientation and operator size biases affect small bodies. Figure 2.26 tests interactions between edges of different orientations. Figure 2.27 tests operator performance in a scene with a wide range of intensity values and edge transition sizes (highlights, shadows, object-object, object-ground interfaces).

For the tests we have implemented, in addition to our two detection schemes, a Hueckel operator [23], a faster Hueckel-like operator [43], a Yakimovsky operator [66] with non-maximum suppression, a Rosenfeld non-linear operator [52][53], and a simple gradient detector with non-maximum suppression [7]. The rules of the game are as follows. Each detector is tuned using the S=0.5 artificial pictures so that performance is acceptable. If not acceptable, at least as good as possible. Then the operator is applied to all the other test pictures using the same confidences and thresholds.

We intend the reader to examine the various edge pictures and draw his own qualitative opinions on the relative merits of the various techniques tested, however we offer some discussion of our results and several quantitative measures (these for the constructed images). Table 2.1 tabulates correct and incorrect edge densities for each operator. Table 2.2 tabulates the length of diffused contour followed by each operator (1.0 if contour is followed across the entire test picture). Table 2.3 tabulates positional variation for edges per unit length along the diffused ramp. Table 2.4 suggests the minimum stripe size an operator can detect without smearing (size in pixels).

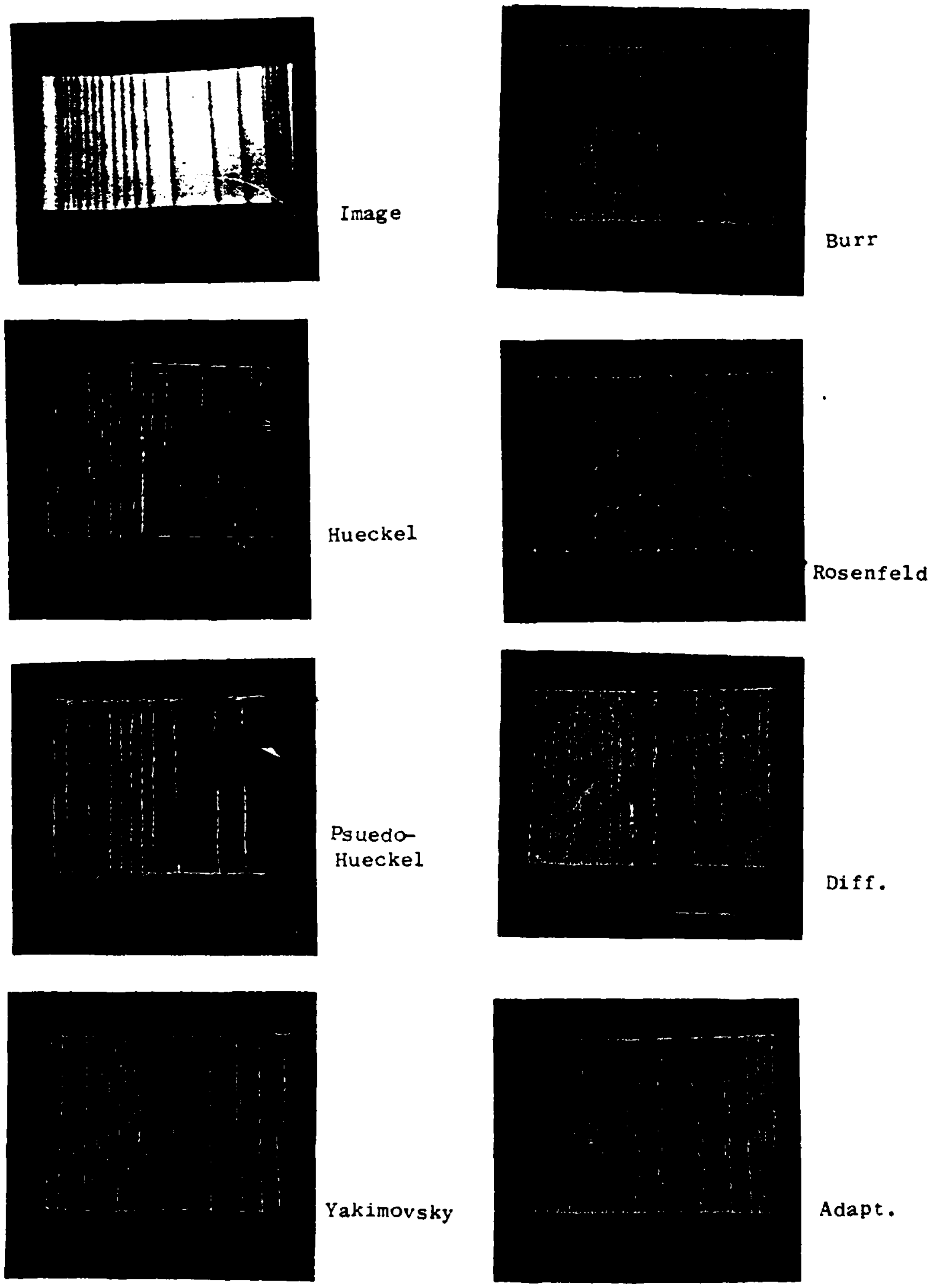


Figure 2.24. Lines

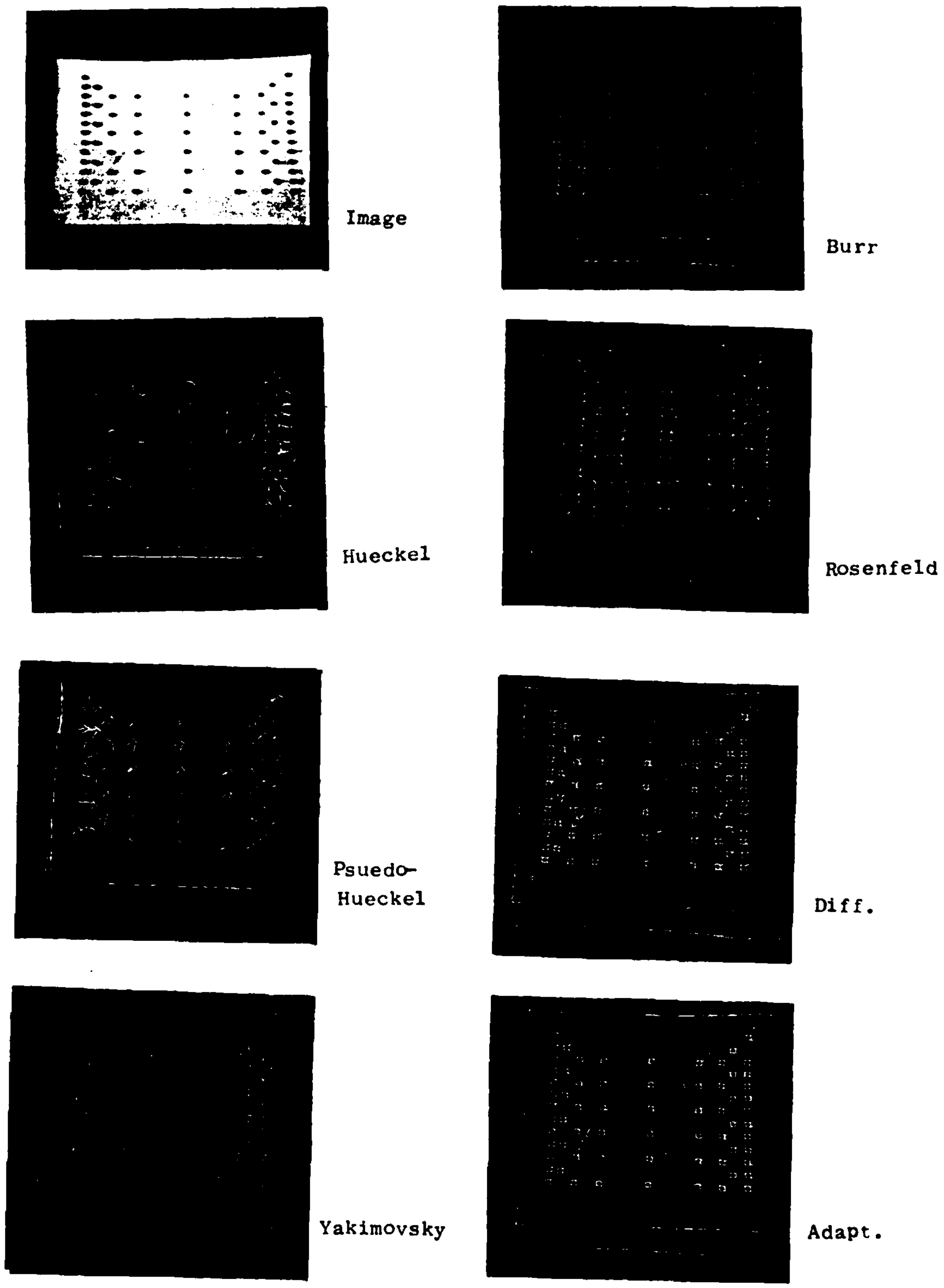


Figure 2.25. Dots

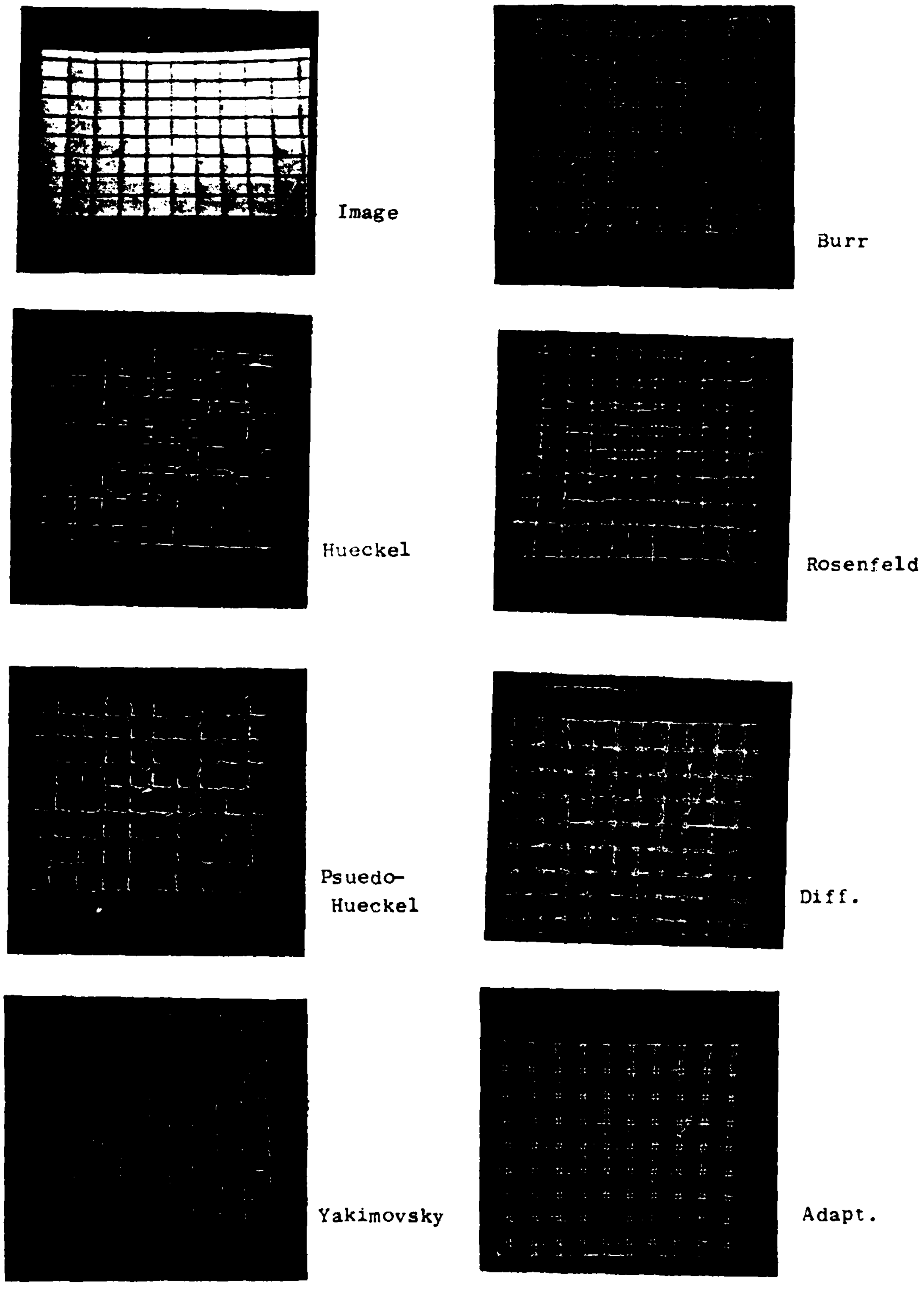


Figure 2.26. Grid

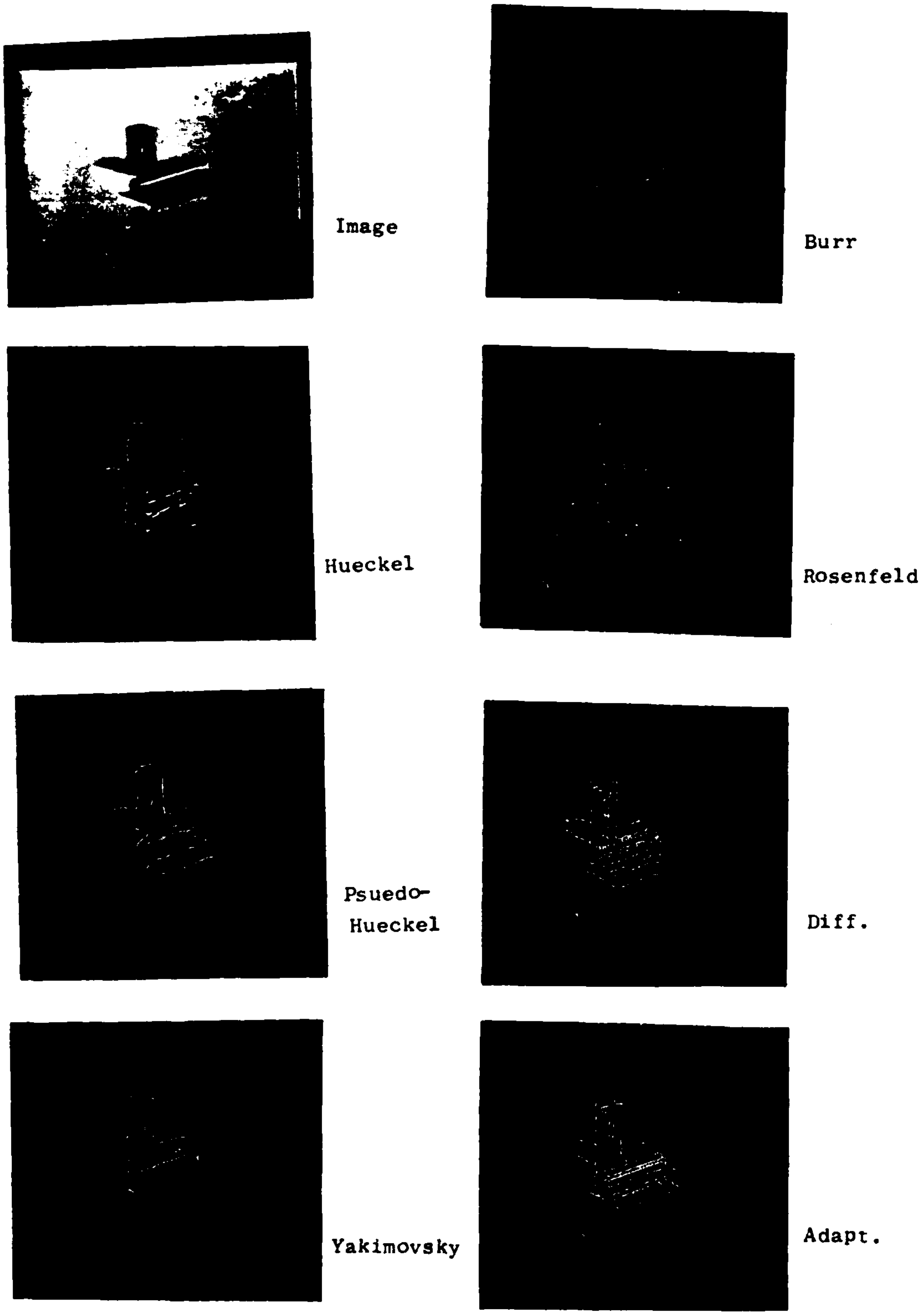


Figure 2.27. Metal Cylinders

Table 2.1. Edge Densities

Operator Name	False Detections Per Unit Area**	True Detections Per Real Edge*
Hueckel	0.03	0.34
Psuedo-H***	0.0005	0.56
Yakimovsky	0.004	0.73
Burr	0.02	0.19
Rosenfeld	0.0005	1.0
Diff.****	0.002	0.97
Adapt.*****	0.03	0.19

* Note that True Detections Per Real Edge also reflects sensitivity of the specified operator to diffused ramps (some operators have trouble following the diffusing ramp very far, therefore score poorly).

** This reflects ambient noise level in the S=1.0 pictures (non-central ramp edges - i.e. edges that should not be detected).

*** Hueckel-like Operator (Appendix A).

**** Difference-based Multiple Operator.

***** Adaptive Threshold Multiple Operator.

Table 2.2. Edge Following Lengths

Operator	Diagonal Length*	Vertical Length
Hueckel	0.2	0.4
Psuedo-H	0.2	0.62
Yakimovsky	0.47	0.77
Burr	0.19	0.19
Rosenfeld	1.0	1.0
Diff.	1.0	1.0
Adapt.	0.19	0.2

* Lengths are scaled by total diffused contour length.

Table 2.3. Positional Variation Along Diffusing Ramp

Operator	Positional Variation*
Hueckel	0.09
Psuedo-H	0.11
Yakimovsky	0.15
Burr	0.20
Rosenfeld	0.09
Diff.	0.00
Adapt.	0.05

* Distance variation orthogonal to diffused ramp scaled by ramp length.

Table 2.4. Minimum Stripe Size

Operator	Stripe Width (Cells)
Hueckel	5-7*
Psuedo-H	5-7*
Yakimovsky	7-9*
Burr	3-4*
Rosenfeld	1-2**
Diff.	1***
Adapt.	3*

* Set by basic neighborhood size.

** Set by smallest neighborhood size. Note, however, due to large operator preference, many times operation in areas with small strips is dictated by the largest neighborhood. In this implementation 16x16.

*** Set by smallest neighborhood size.

2.6. Sensitivity, Noise Immunity, and Shape Deformation

If by sensitivity, we mean ability to detect gradual gray scale transitions, clearly the Rosenfeld, Yakimovsky, and our difference based operators are the best. The Yakimovsky does well because of a large neighborhood size. This will be a draw back in detecting smaller features in scene. Both our operator and Rosenfeld's do well in the domain of diffused lighting because they both have the ability to average over large areas, thus detecting small gradual changes. The Hueckel and Hueckel-like operators are a disappointment. With such large neighborhoods (9x9) one would hope for better gradual slope detection. Possibly, this can be explained by considering that these operators are quite finely tuned to detect step changes. The gradual slope generate too low a confidence step.

In the real pictures, the results with respect to sensitivity seem to corroborate those from the artificial images. The Hueckel and Hueckel-like operators are universally less sensitive than the various difference-based operators (Diff., Adapt., Rosenfeld, and Burr). In the line picture, many lines detected are single (instead of paired), and miss directed segments are common. In dots, the dot shapes are badly smeared and deformed. Directional interactions in the grid picture seem to occur at each vertex. These operators do the best in the metal cylinders, but still not better than the other techniques.

The Yakimovsky operator, on real pictures, tends to smear small detail as is to be expected with such a large neighborhood operator. In lines, double lines become single lines. In the dots shot, each dot is enlarged and flared at the corners. (Corners on a round dot? See for yourself.) These flaring corners come from the peculiar neighborhood shapes proposed for this operator [66]. The grid shows the same signs as the lines picture (singles where there should be doubles). In the metal cylinder picture, we get a good outline but are weak on internal details and shadow completions. Yakimovsky's operator is better than the Hueckel and Hueckel-like operators, but is less effective than the difference-based techniques.

The Rosenfeld non-linear operator does very well for somethings and poorly for others. On gradual vertical ramp detection, it gets nearly a perfect score. For diagonal following, as the ramp gets further diffused, edge position varies wildly. This occurs in part due to our use of only two orientations. The diagonal orientation causes the vertical edge peaks to be displaced from the horizontal peaks due to the square shape of the averaging windows used in differencing. More orientations in the detector system could help. Also narrow (non-square) averaging rectangles would minimize the problem.

In the real picture the Rosenfeld operator does poorly because its large operator preference causes it to lock on lighting (metal cylinders) effects or line density (lines) effects. We have not included textured pictures as in [52], though if we had this operator would have performed well on them. Basically we see a common idea being applied to two different problems, getting rather different results. The Rosenfeld strategy is to bias a multiple template detection scheme towards the large operators, to average out textures. Our difference-based technique is biased towards the small operators which are significant, to detect all fine structure without undue smearing.

The Burr techniques and our difference-based technique are very similar. Burr's detector has the same shape and size as our detector number 3 (Figure 2.5). We used center of mass or absolute maximum peak selection, while Burr uses relative maximum. This explains Burr's lower noise immunity. Being a single, small neighborhood, Burr's detector has problems when a edge becomes to diffused. This shows in the ramp pictures and later in the metal cylinders picture (Note the left most shadow in our difference-based output and Burr's. He misses edge points critical in completing the shadow contour because they are too diffused and weak).

Our difference-based operator does well on all the test pictures. The position difference between vertical and horizontal edges in tne diagonal test pictures occurs because on diffused edges, large operators are required. These operators fall off the end of the picture if the ramp is too close the border. The cells off the picture are set to the value of the last cell on the border of the picture. This is not a good heuristic. Therefore, the positional inaccuracy is not due to the basic operator itself. We should not be operating that close to the edge of the picture with large operators.

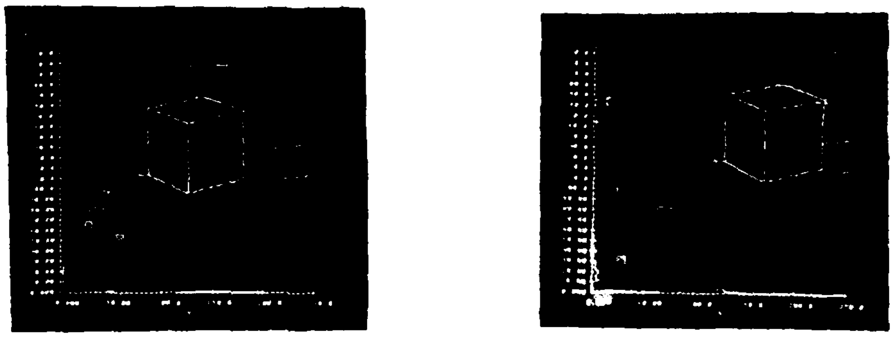
Our adaptive threshold technique is about at the same level of sensitivity as the Burr operator. This is because Burr subtracts cells in multiples of threes and so does this implementation of the adaptive operator. This operator has less noise immunity due to the additive Laplacian (these operation are actually digitized second derivatives, therefore high frequency and noise enhancers). On the other hand, it is more sensitive in real pictures (note extra definition in the lower cylinder of the metal cylinders picture).

We feel the difference-based operator, for an application requiring the detection of both fine detail and diffused (shadow) contours, is the overall winner. The only qualification is that the user must be able to select an accurate noise model (set of thresholds for minimum significance). Without this, the results go down under the noise easily.

2.7. Conclusions

The overall results of this study indicate that first, difference base edge detection is not only the easiest to implement, most efficient to run, but also in most cases the best, when coupled to a reasonable edge peak selection algorithm. Second, peak selection via a "global" method (such as center of mass or absolute maximum/minimum) is superior to a "relative" method (relative maxima, non-maxima suppression). Thirdly, several edge detector sizes are superior to one because small ones can adapt to fine structure and larger ones can pick out harder to "see" gradual intensity changes. Stable Higher-level processing is based on solid, accurate low-level measurement (Figure 2.28).

Images (Right - Left)



After Difference-Based Edge Detection

After Supressing Disconnected Edges
And Forming Regions

Right And Left Retina Images, Edges, And
Edges Derived From Region Boundaries
Figure 2.28

Chapter 3

Intermediate Vision - Vertex-String-Surface Graphs (V-S-S)

3.1. Introduction

We will introduce novel algorithms for region (surface) aggregation, boundary string following, and vertex detection and reconstruction. In our view, visual recognition can be divided into three distinct levels. The first is basic measurement, in this system edge detection and image sampling (Chapter 2). A system will be fundamentally limited in discrimination power by the dimensions of measurement chosen. We have limited ourselves to a system using black and white or color images (we process texture in a limited way), with all pertinent image changes characterized as steps (the position, orientation, and transition size of each step is recorded). Image dimensions are variable from 1x1x1 bit to 504x476x9 bits, however we typically have used 252x238x5 bits (edge detection experiments in Chapter 2) or 252x238x9 bits (following chapters).

The second phase is object-directed segmentation, or intermediate vision. In this phase, we combine basic measurements into primitive groupings representing major object components. The object components generated may be considered as objects in their own right, however differ from learned objects because their interpretation is fixed. Basic measurements have preprogrammed form and preprogrammed interpretation (such as intensity steps, spots, color feature vectors, intensities). Object components have preprogrammed meanings, but variable forms (regions - made up of an arbitrary collection of adjacent feature and position measurements, boundary strings - made up of a collection of adjacent intensity step points, vertices - made at the coincidence of an arbitrary number of strings endpoints).

The third phase is object formation and recognition. This will be dealt with later (Chapters 4 and 5). Learned objects have variable forms and variable meanings. In practical vision systems, there are no such things as impossible objects [25]. All objects that are formed by data measured from the real world are possible and must be represented.

Our system for intermediate vision reads edge data, transforms this to edge and region data (still in pixel format). These two forms of information may be passed to texture processors or shape processors. Texture processes transform edge information (masked by region data) into smoothed images for further edge detection. Shape processors superimpose edge data from multiple sources (region differentiations, intensity edges, colored image edges, or texture image derived edges), and extract contours and vertices in the form of the vertex-string-surface graph structure (V-S-S). Further scene to scene matching and recognition is done with data encoded in these V-S-S structures alone.

3.2. Related Work

Compared to work in low-level vision, relatively little work has been done in intermediate vision. The work of Marr on occlusion, depth extraction, and texture is significant [31]. Motivated by neurophysiological results Marr proposes measurements in the form of a primal sketch, basically a map of edges. The measurements in the primal sketch are then correlated together for various texture parameters, contours, symmetries, and depth primitives. The proposed texture model is based on histograms of edges and connections between near edges (histograms of either directions of edges and connections or sizes of edges and connections). The critical idea in Marr's work is that vision consists of forming symbolic descriptions of visual forms at every level of representation, not simply at the level of object descriptions. In this way Marr's ideas and ours about object components are very similar even though, in form and process, they are somewhat different (while again proposing a different computational attack, Waltz also shares this affinity towards symbolic low-level vision).

Some work has been done on region growing algorithms which incorporate some concept of figure "goodness". The Brice and Fennema region grower merged to minimize perimeter growth, therefore favoring compact regions [6]. The region-based system of Tenenbaum used a priori information to partially control region merging [60]. Feldman and Yakimovsky also have pursued region-based techniques which incorporate semantics, via a Baysian strategy [14].

In the area of contour extraction and codification very many ideas have been examined. Martelli uses an optimization technique, coupled with potentially extensive search to find optimal contours [40]. Very many chain encoded contour techniques have been used. Techniques based on Hough transforms have been used both in domains consisting exclusively of straight line segments and those having limited, but more general segment types [46].

Corner detection in most cases, has been done by inference from contour data [16][47]. Some work has been done on direct corner detection [12]. We have found that any attempt to directly detect corners in image data is relatively insensitive. In both corner detection and contour codification, the idea of curvature arises. Curvature information extracted from a fixed grid system is both a function of the underlying curve and the image sampling system. This interaction has been studied for some pixel spaces [5].

From a computational point of view, the most common algorithms at the intermediate level have been simple and sequential [48]. However, some have used searches with backup [58]. Much recent interest has been shown in relaxation algorithms [54]. Primarily those with probabilistic interactions have been used. Examples of symbolic relaxation have been employed at a higher level [63].

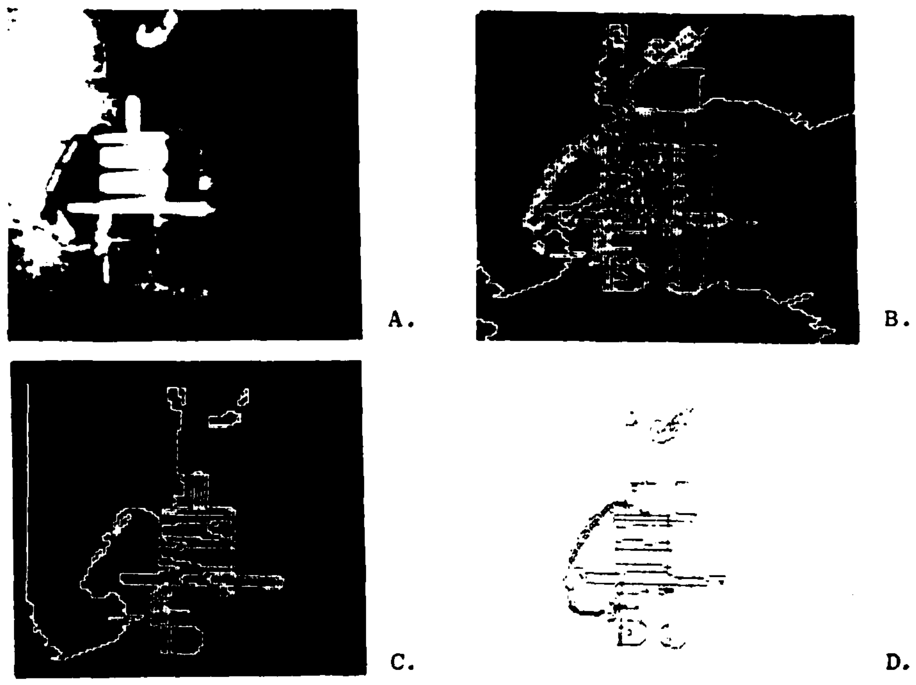
We will first examine region formation from edge point data. We will then examine our method for integrating edge data from several sources with region data for the boundary-vertex extraction process. V-S-S graph formation will be discussed, followed by some notes on multi-color processing and texture processing.

3.3. Regions

Artifact vision is object oriented. Objects may be viewed as collections of surfaces which enclose volume. Each of these surfaces project onto the image space as one or more two dimensional regions, each having relatively homogeneous characteristics. Most commonly observed objects have surfaces which are compact (convex-connected), or have surfaces which are easily decomposed into compact subsurfaces. This indicates that our algorithms for region aggregation on the image space should, whenever possible, be biased in favor of compactness.

Biological vision systems seem to be sensitive to figure-ground relationships. By this, we mean that regions in the two dimensional image space can be ordered by relative embedding relationships, which correspond to object level-background level separations. Objects are figures with respect to the actual background. Painted surface details are figures with respect to major object surfaces. Highlights are bright figures with respect to surface details or object surfaces. These embedding relationships can sometimes be thought of as links between levels in the dendograms generated by a traditional intensity region merging sequence [29]. In general however, these relations are purely geometrical. For object component region generation embedding relations should be recovered, and used in the region growing scheme if possible.

Strict intensity-based region growing schemes have a tendency to either generate large numbers of regions when severe thresholds are used, or a smaller number of regions, where each actually corresponds to more than one object surface, when loose thresholds are allowed (Figure 3.1b-c). In fact, places where several object surfaces tend to merge in region-based processing, are also places where edge detection systems fail to find high confidence edge points (Figure 3.1d).



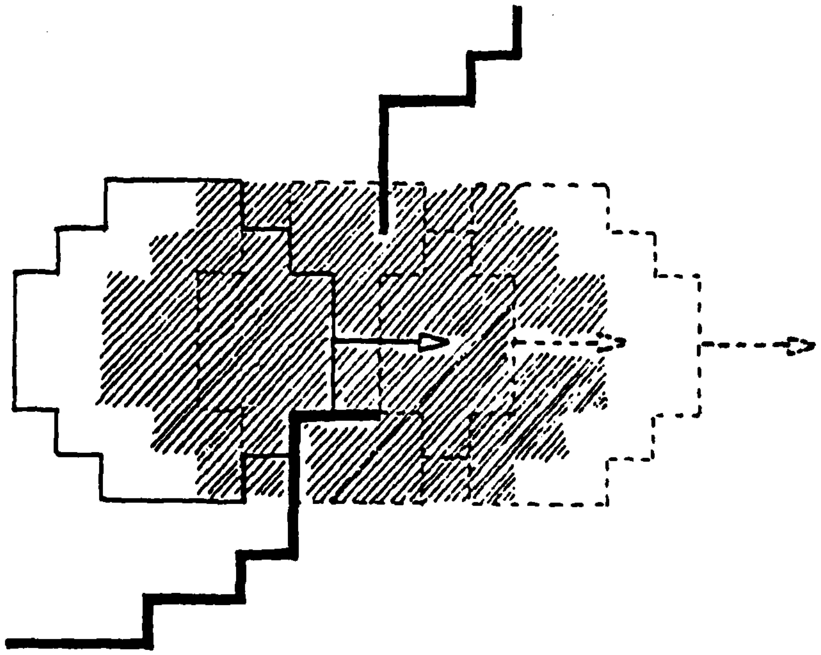
A) Sample Scene;
B) Regions From Strict Threshold System;
C) Regions From Loose Threshold System;
D) High Confidence Edge Points.
Figure 3.1

To remedy the problem of low confidence local data, some form of a priori information needs to be incorporated. Sometimes scene related information has been used, however, this restricts generality from scene to scene. We have chosen to build our region aggregation system around the idea of figure "goodness". That is, we make regions which tend to be locally compact.

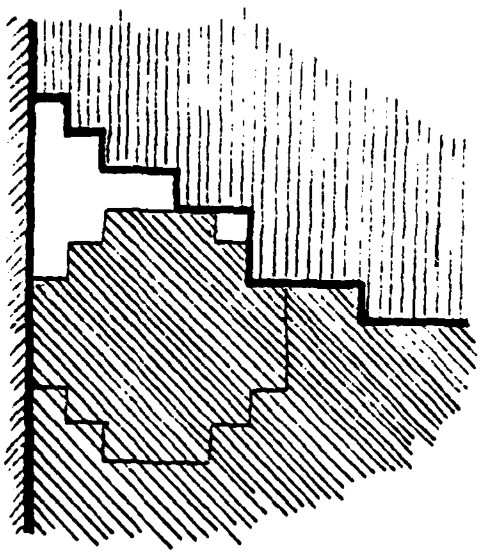
3.4. Homogeneity

We have defined homogeneity in a domain or area to mean absolute absence of high confidence edge points. In short, our region aggregation scheme consists of assigning unique region numbers to each connected homogeneous area. All non-homogeneous areas are then classified by several schemes to refine the shapes of the homogeneous regions.

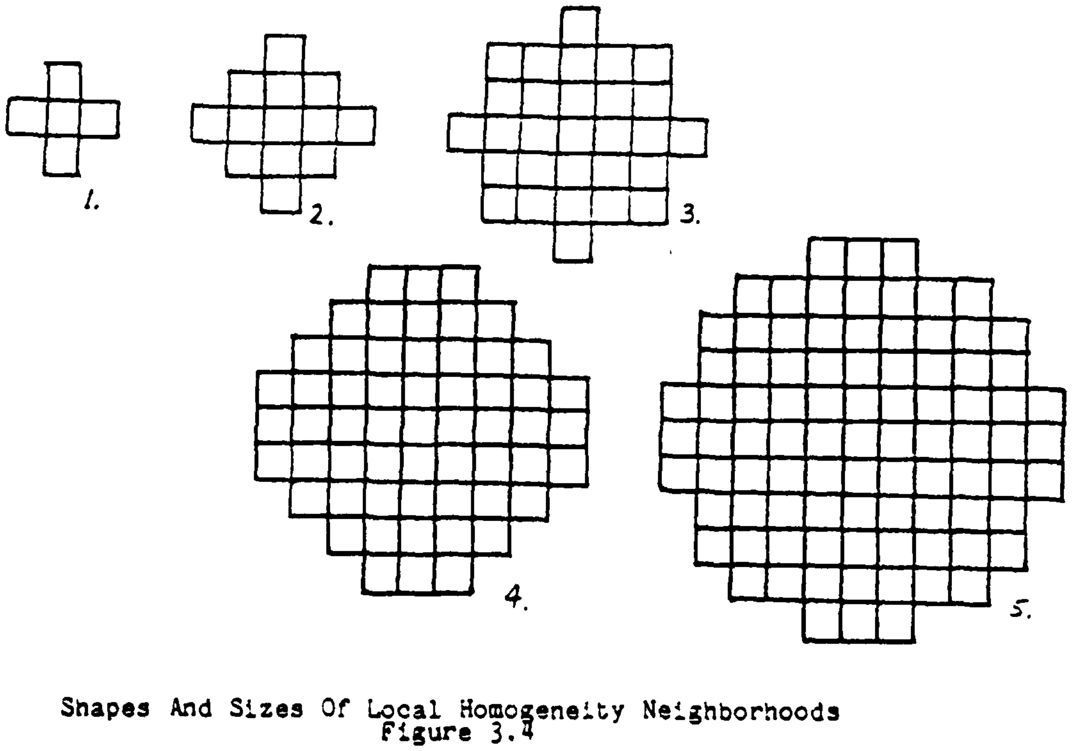
The larger the local domain used for homogeneity computations, the larger the length of missing edge data that can be completed (Figure 3.2). However, the larger the operator size, the more sharp curvatures are modified (Figure 3.3). We use digitized approximations to disks for the basic homogeneity operations for computational ease and to minimize boundary shape biases (Figure 3.4).



Homogeneity Operator Passing Through A
Break In Edge Data
Figure 3.2

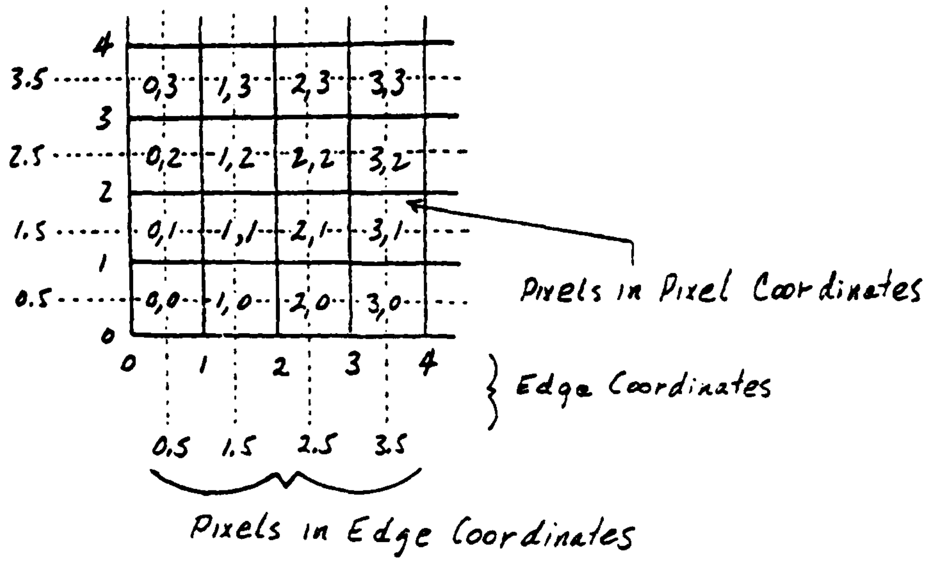


Area Of Non-Homogeneity Around A Corner
Figure 3.3



Shapes And Sizes Of Local Homogeneity Neighborhoods
Figure 3.4

Note the arrangement of pixels and edge points in Figure 3.5. Edges come in two kinds, vertical and horizontal, each between pixels. The edge positions inside the domain boundaries are checked in the homogeneity computations.

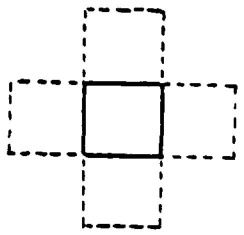


Edge and Pixel Coordinates
Figure 3.5

3.5. Region Sweeping

To form regions, we sweep a homogeneity operator (one from the set in Figure 3.4), uniformly over the image space. As edge points enter the domain, a counter is incremented. As they leave, the same counter is decremented. The cells in the homogeneity disk are marked (set to a region number) if the counter is zero (i.e. centered on an area including no edges). If the counter is non-zero, the next homogeneity position is evaluated. As an efficiency, the image space is divided into 32x32 sectors. After all the cells in a sector have been marked, a bit in the corresponding location in a 32x32 map is set, preventing further evaluations within the sector.

The marking operation also involves generation of region numbers (these numbers are the marks, recorded in a "region" image, where each pixel contains an index into the region property table). There are three methods used for generating numbers. The easiest is used if none of the four cells adjacent to the central cell has been assigned a number (Figure 3.6 - We do not consider cells on the diagonal to be adjacent because there is not a shared edge position). In this case, the next sequential number (and region table entry) is assigned.



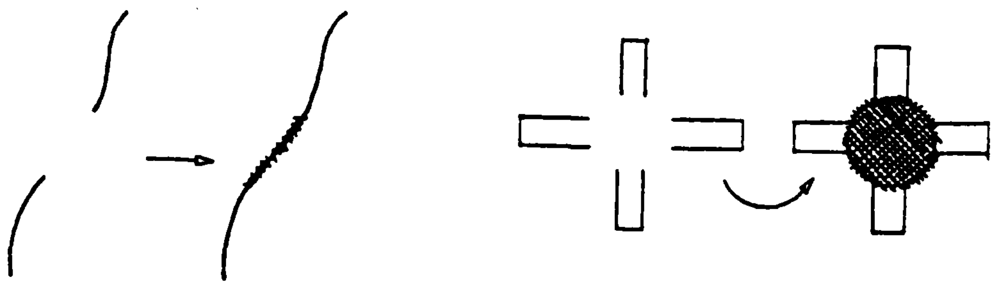
Cell Adjacency
Figure 3.6

The second easiest case occurs when only one region number occurs in any or all of the adjacent cells. In this case, the region number assigned is this number (thus smearing the region along). The difficult case occurs when there are several different numbers from which to choose. In this last case, all the regions indexed that have been formed previously in this sweep (or pass), are "merged". The merging consists of selecting one of the indices as a root index (this becomes the index to be used for new marking), and modifying the table entries for all the other region indices to "point" to the root entry. Therefore all the non-root entries become "aliases" for the root number. In this way we avoid renumbering previously marked cells.

If some of the region indices adjacent to the central cell represent regions formed in a sweep other than the one currently being performed, simple merging would not always be appropriate (next section). In these cases, an association is stored between the old region entries (from prior sweeps) and the new one (formed this sweep). When a single pass is finished (an operator has been evaluated over every unmarked cell), these associations are examined.

It should be noted that region number propagation occurs through only the directly adjacent cells, however, cell setting occurs over the entire homogeneity disk area. This little trick makes it necessary for any edge string breaks to be on the order of the diameter of the homogeneity disk before the two different regions are jointed. Also it causes the remote possibility of setting noncontiguous areas to the same region numbers.

The sweeping operation occurs for each different operator size from the largest to the smallest. Following each sweep pass, region table space is reclaimed and cells are reindexed ("aliases" are compressed out). The larger the operator allowed to aggregate a given region, the larger the breaks in edge data that can be tolerated. Also, some classes of subjective contours are generated because of compact area enclosure (Figure 3.7).



Subjective and Real Contour Completions
Figure 3.7

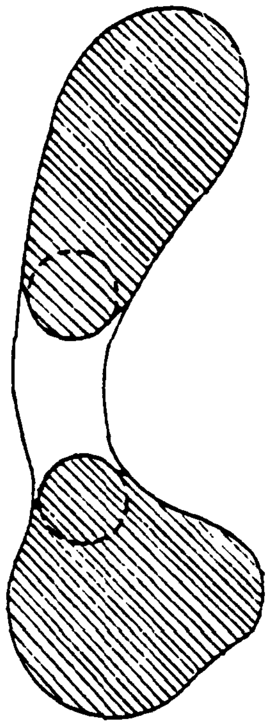
3.6. Region Noise, Ambiguity, and Corners

There are some striking similarities between this scheme for region aggregation and the edge detection techniques previously discussed (Chapter 2). In the edge detection scheme, small detectors control the process. In the region system, larger operators play the same role. In the edge detection scheme, small operator outputs are evaluated against a noise model to determine their significance. Similarly, the regions formed after a sweep operation need to be checked for significance. This is done in two ways. The first and most obvious requires regions to be of significant size. The size parameter we have used in our experiments is nominally 50 cells (out of 252x238). This significance dimension tends to become important for sweeps with smaller operators (size 3,2, or 1).

A slightly less obvious significance parameter arises from the relative sizes of the homogeneity operator and the region aggregated by this operator. If the operator is too large (diameter approximately the same as the smallest body crossing chord), one region may be broken into several parts (Figure 3.8). It is easier to get such things right than to patch them up later so we have defined the following parameter:

3.6.1. $\rho = (4 * \text{area}/\text{perimeter})/(\text{size} * 2)$

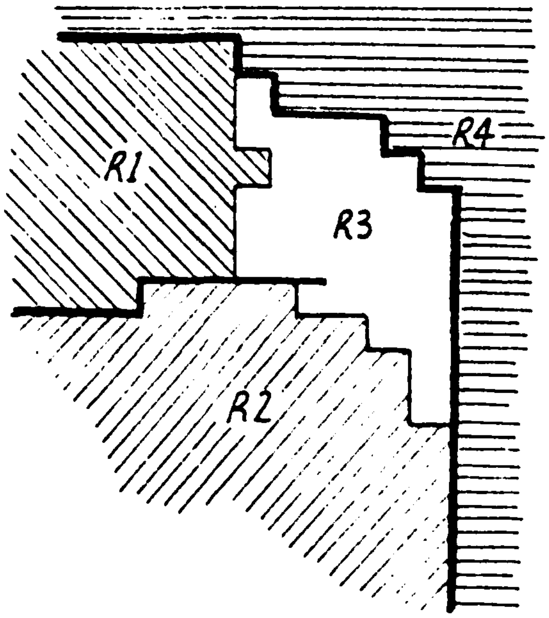
If this ratio is greater than a threshold, the estimated region diameter (4*area/perimeter) is significantly greater than the operator diameter (size*2), and therefore this region is significant. This significance test tends to eliminate undesirable large operator aggregated regions. The threshold we use is nominally 1.0.



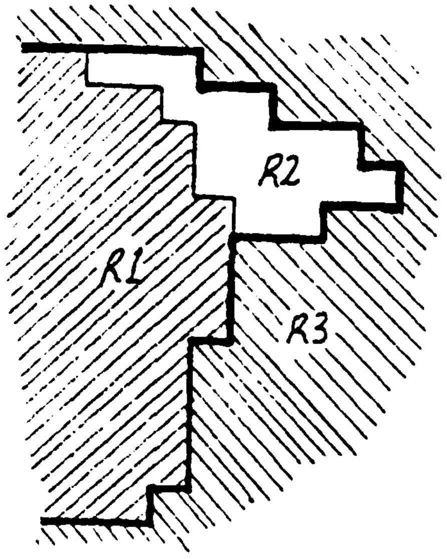
A Region Broken By A Large Homogeneity Operator
Figure 3.8

Following each region generation sweep, during region table space compression, we have the opportunity to reassign region cells to "unprocessed". If a region is computed to be insignificant, its cells are so marked, and the corresponding entry in the region tables is eliminated.

If sharp curvatures exist in the bounding contours of a region, there will be areas at these corners where large operators will fail. Successively smaller operators will be able to succeed in such areas. However, the smaller operators will tend to give incorrect results due to the fact that near these corners high confidence edges will be absent (Figure 3.9). It will be generally difficult to determine where the small regions at corners should be incorporated. We would merge them into an adjacent region if such a merger was unique (i.e. the small region is associated to only one other region - Figure 3.10), but when uniqueness is not met the ambiguity is not easily resolved. In this ambiguous case cells in the region are marked as ambiguous, and left to be processed in the final refinement passes following the sweep of the homogeneity operator.



Small Region Near A Corner With Ambiguous Possible Merges
Figure 3.9



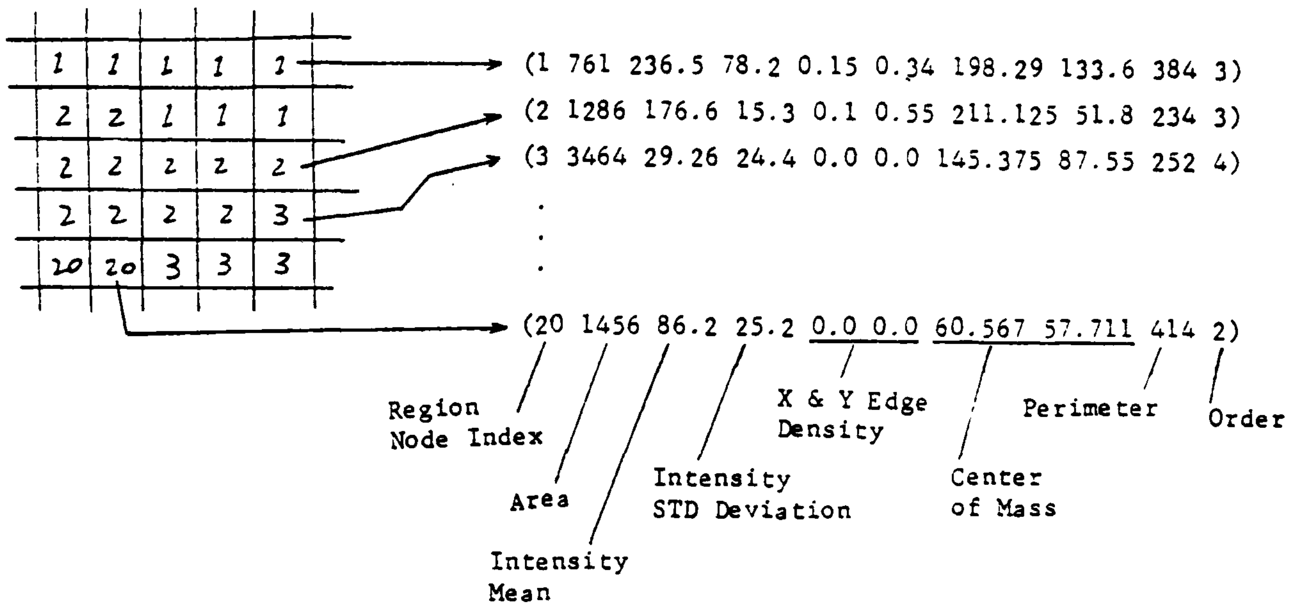
Small Region Near A Corner With
Only One Possible Merge
Figure 3.10

By now the reader has probably realized that the post pass following each region sweep is actually a bundle of kludges (more politely, heuristics), designed to take care of some of the more pathological problems posed by the homogeneity operations. The last operation done in this post pass is to kill any small group of cells set to a particular region number, but not connected to the main mass of that region. Recall that this defect comes about from the region setting and propagation method. An advantage of simple convex shapes like disks is that the problem of these noncontiguous sets is minimized.

3.7. Final Region Refinement

After sweeping with all available homogeneity operators, some cells will either be unmarked (near some region bounding contours) or marked as ambiguous. We want all cells classified as part of a region, therefore we have included a series of passes which assign cells adjacent to established regions to the region which the cells are nearest to in intensity space. This is similar to traditional merging, intensity-based region growing, with the exception that no new regions are formed. Usually no more that 15 passes are required on 5 percent of the image space (due to the sector marking scheme), or approximately 15 seconds of processing (quite fast compared to region generation). This intensity growing algorithm is capable of recovering contours near vertices quite well, certainly better that the original edge detectors.

The final result of the region system is a picture of region indices and a list of regions and region properties (Figure 3.11). We accumulate area, perimeter, centers of mass, intensity average, intensity standard deviation, roughness, and order (operator size which formed the region). Regions are analyzed to determine their relative levels of embedding. We are primarily concerned with level-0 embedding which represents the main background (such as a table top or floor), level-1 embedding which represents all object details (this is all levels between level-0 and level-2), and level-2 embedding which represents highlights and the innermost surface details (this level may not be present, and may not be tangent to level-0 regions). Figure 2.28 shows two views of a block (A), the edges detected (B), and the boundaries which correspond to region differences (C).



```
(1 761 236.5 78.2 0.15 0.34 198.29 133.6 384 3)
(2 1286 176.6 15.3 0.1 0.55 211.125 51.8 234 3)
(3 3464 29.26 24.4 0.0 0.0 145.375 87.55 252 4)
.
.
.
(20 1456 86.2 25.2 0.0 0.0 60.567 57.711 414 2)
```

Region Token and List Structure
Figure 3.11

3.8. Image, Edge, Region Data Merging

The construction of boundaries and vertices can be divided into three main processes. The first is merging input edge data from several sources: the original gray scale image, the edge detection data, the region maps, and texture processors (after edge detection on texture smoothed images). The second is actual boundary and vertex codification. Last is smoothing and reorganization of data into the final vertex-string-surface (V-S-S) graph structure.

The merge process reads the image, the edge images (these have one edge position for each pixel interface in the original image), and the region image to produce an edge list. Each point in the output edge list may be accessed directly by its original position on the retinal space by a simple array reference (each Y coordinate has an array slot pointing to an X coordinate sorted list of endpoints) followed by a binary chop (locates the edge in a selected X coordinate sorted edge list). The region data is used as a template for edge data. Any position in the region picture where region numbers differ, an edge is indicated. If a real edge point exists near this position (i.e. is marked near by orthogonally from the corresponding position in the edge images), an edge is entered into the output lists as a region separator edge (region image indicates two separate regions) and inherits the properties of the real edge. If no real edge is near the region separation, a subjective edge is entered into to output lists. In this way, all region separator edges are recorded insuring completely bounded regions.

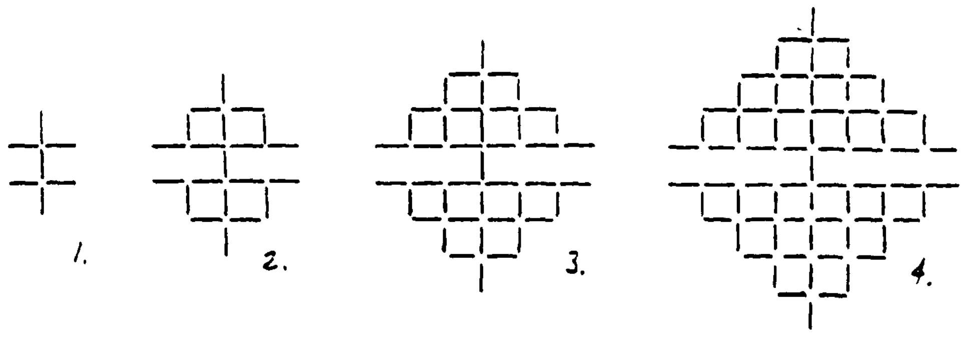
Any real edge points marked, in the edge images, but not absorbed by region separations are output as non-separator, real edges in the output lists. These can correspond to incomplete boundaries, or internal creases on a surface. Non-separator edges are more suspect than separator edges as possible noise points. Subjective, separator edges are more suspect in positional information, because the region aggregation process simply put them where it wanted them.

The gray scale image is used to compute the intensity average immediately on both sides of edge points. The intensity is sampled at 1/2 the transition size of the edge point on both sides of the edge. Subjective edges are taken to have a transition size of zero.

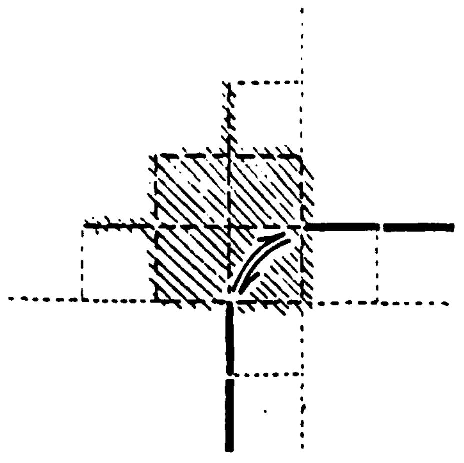
3.9. Vertex Detection and Boundary Codification

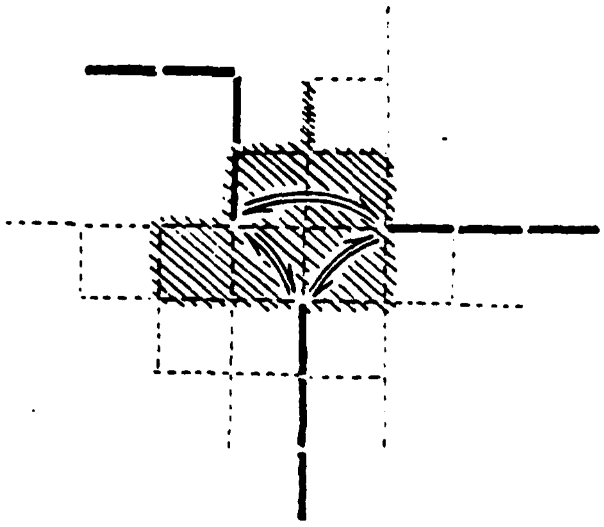
Vertices are really discontinuities in several dimensions. They are discontinuities in image intensity (as are edge points). They are discontinuities in direction (i.e. sharp boundary inflections), and in topology (separating two or more regions, two or more strings). By string we mean "macro" edge points. That is, chains of edges which connect end to end and have only two end points (terminating in vertices, looping back to each other, or terminating in free space). To find vertices we really want to find points where all the above discontinuities coexist.

First we attempt to attach each edge endpoint to another edge endpoint in a reflexive manner. By this we mean, if endpoint A can only be connected to endpoint B, and endpoint B can only be connected to endpoint A, they can be connected. A connection neighborhood is expanded about each edge endpoint as in Figure 3.12. Possible edge connections are checked at each order of expansion. When reinforcing connections as in Figure 3.13, are found, reflexive linking occurs. Edge endpoints involved in more complicated connection geometries are flagged as possible entries into vertices. A new vertex number is assigned for each of these areas of connection ambiguity. In effect, these possible vertices are detected where topological discontinuity is locally indicated.



Connection Neighborhoods, By Order
Figure 3.12





Reinforcing and Nonreinforcing Connections
Figure 3.13

After all reflexive linkages are made, strings may be assigned by finding edges marked as entering vertices at one end and reflexively linked chains on the other end. The reflexive links are followed to find the successive edges to be bound in the string. Circularly linked chains will still be left. These are broken by artificial vertices and processed in the same manner as non-circular chains.

Some vertex clusters will be simple groups of edge points, with no string connections, or 0-degree vertices. Some will be 1-degree, with only one string connection. These types have no topological significance and can be discarded. 2-degree vertices can be either noise generated or real. Some that are real will not be detected by the simple topological discontinuity criterion. For this reason, 2-degree vertices are also discarded, to be regenerated at points of high curvature later. The appropriate string concatenations are then performed. The vertices that remain are 3-degree and higher, and relatively certain, but their positions are not accurately known.

3.10. Vertex Reconstruction and String Smoothing

Reconstruction and smoothing is a two pass operation. We first compute accurate positions for 3-degree and higher vertices. We then smooth and detect 2-degree vertices at points of maximum curvature. The reconstruction and smoothing is then performed again using all vertices.

Vertex reconstruction consists of computing the perpendicular least-squares line fit* for each string entering the vertex.

3.9.1. $$\bar{x} = \frac{1}{n}\Sigma x_i \quad \bar{y} = \frac{1}{n}\Sigma y_i \quad \overline{x^2} = \frac{1}{n}\Sigma x_i^2$$

$$\overline{y^2} = \frac{1}{n}\Sigma y_i^2 \quad \overline{xy} = \frac{1}{n}\Sigma x_i y_i \quad (x_i, y_i) = \text{Edge Point Coordinates}$$

$$S = \overline{xy} - \bar{x}\bar{y} \qquad n = \text{Number of Edge Points}$$

$$cc = (\overline{x^2} - \overline{y^2}) - (\bar{x}^2 - \bar{y}^2)$$

$$T = -cc/\sqrt{cc^2 + 4S^2}$$

$$A = \sqrt{1+T} \quad B = [-\text{sgn}\, S]\sqrt{1-T} \quad C = A\bar{x} + B\bar{y}$$

for line: $$Ax + By = C$$

Two linearly independent equations are all that are required to position a vertex. When more are available (3-degree vertex or more), the system is solved for the best intersection point in a least-squares sense.

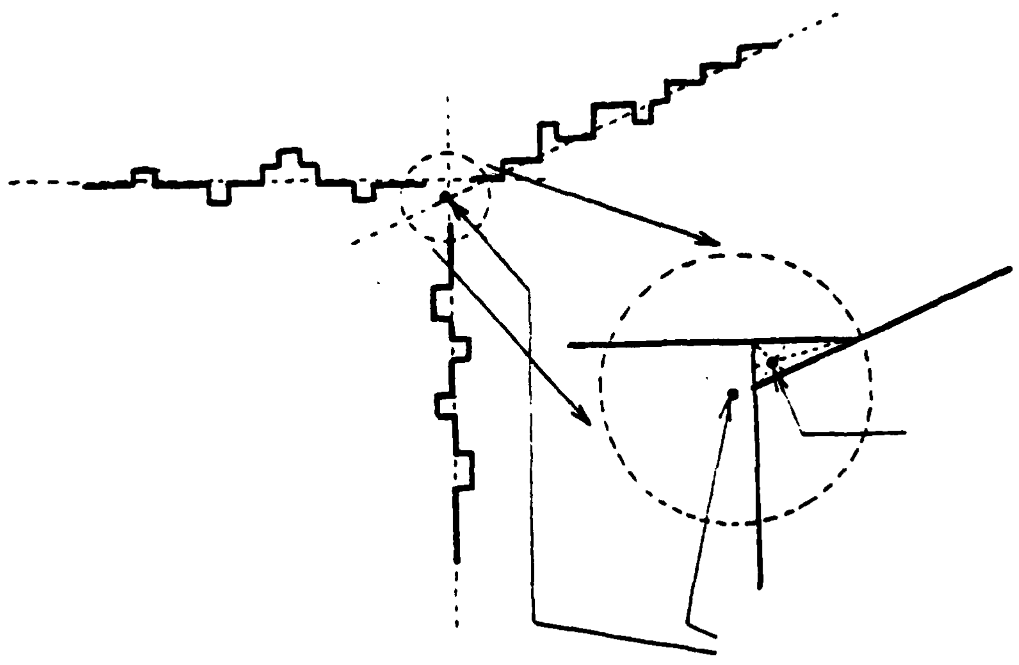
* These equations (3.9.1) and several others are courtesy of Rod Fletcher.

3.9.2.
$$\begin{aligned}
[A_1 x + B_1 y &= C_1]\ \omega_1 \\
[A_2 x + B_2 y &= C_2]\ \omega_2 \\
&\vdots \\
[A_n x + B_n y &= C_n]\ \omega_n
\end{aligned}$$

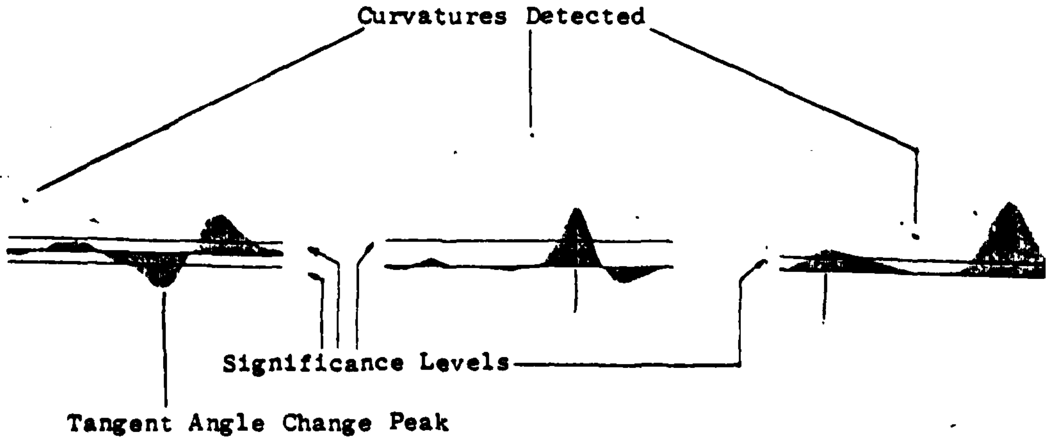
n equations n weights

Solve:
$$\begin{bmatrix} \sum\limits_{i=1,n} \omega_i A_i^2 & \sum\limits_{i=1,n} \omega_i A_i B_i \\ \sum\limits_{i=1,n} \omega_i A_i B_i & \sum\limits_{i=1,n} \omega_i B_i^2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \sum\limits_{i=1,n} \omega_i A_i C_i \\ \sum\limits_{i=1,n} \omega_i B_i C_i \end{bmatrix}$$

When vertex positions are fixed, a least squares line is fitted at each point in the strings (Figure 3.14). The A and B coefficients are used to compute a tangent angle. These angles are used to compute change in angle with respect to arc length, or curvature along the strings. Two degree vertices are recovered by examining these curvatures. As in linear difference edge detection, an edge (in this case a 2-degree vertex) is located at points where the difference (curvature) is greater than a threshold T and maximum over a specified domain (Figure 3.15). All the tricks used for edge detection are applicable in this situation when scaled down to a single dimension (along the string). In practice, we may use a single sized difference operator with a single threshold and get acceptable results.

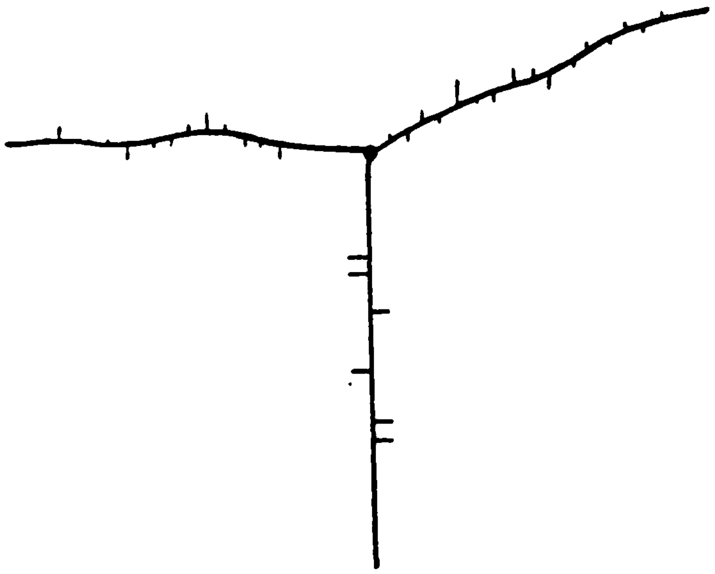


Vertex Reconstruction
Figure 3.14

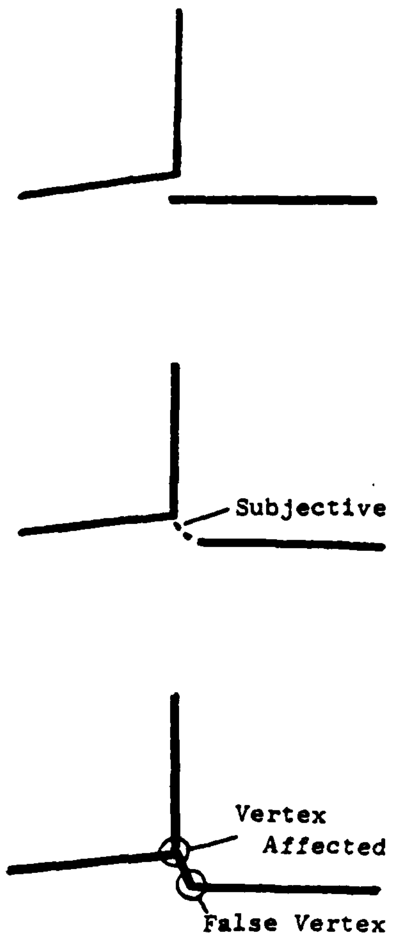


Detection Of Significant Curvature Discontinuities
Figure 3.15

After the 2-degree vertices are marked, we perform the the same reconstruction algorithm on them, followed again by the linear equation fitting about each string point. This time however, each string point is repositioned along the perpendicular to the least squares line onto that line, thus smoothed (Figure 3.16).


Edges Along A String Moved During Smoothing
Figure 3.16

Sometimes, incorrect highly subjective edge strings cause false vertices to form, perturbing the positions of nearby real vertices (due to our perhaps too powerful smoothing and vertex repositioning machinery - Figure 3.17). This is corrected by a post pass which examines all edge strings that are highly subjective (ones formed solely by our region aggregation system, without adequate edge data corroboration). When required, these highly subjective strings are deleted, and the vertex positions in the vicinity are recomputed. The region data base remains unchanged by these deletions, therefore some regions may only be partially surrounded by contour data. The missing sections correspond to areas where connection is assumed, but where the exact nature of that connection is in question.

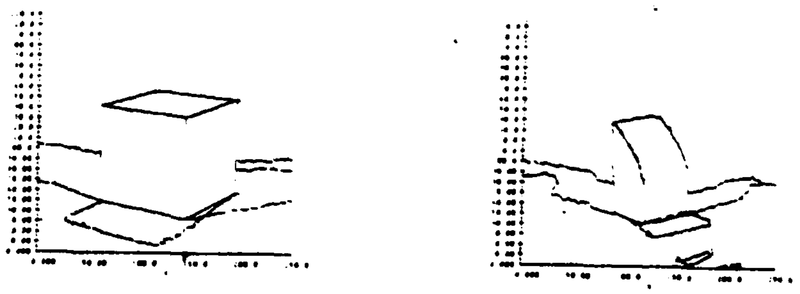


Subjective Contour Forming A False Vertex
Figure 3.17

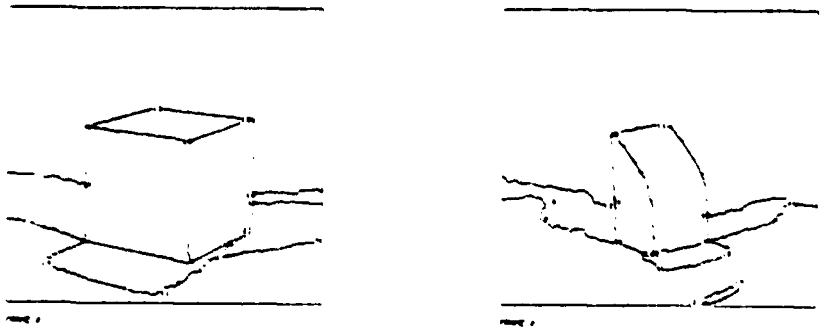
At this point we now have a data base containing all vertices linked to the strings which form them, along with all entry angles. We have the strings as property lists containing the average transition sizes of their edges, the average intensities to the left and right of their edges, the variations in intensity to the left and right of their edges, the variations in the differences across their edges, the regions separated by their edges, and indices to ordered lists of edge positions and tangents, to describe the string shapes. Lastly, we have a region list, containing the properties computed by the region aggregation sub-system.

From this point forward, all processing is removed from the original picture space. All successive operations are performed on the vertex-string-surface graph structure. Figures 3.18-3.24 show the images, images after edge detection, edges after vertex marking (X's mark edges entering possible vertices), vertex initial and reconstructed positions, distance moved by edges during smoothing, the V-S-S structure, and the V-S-S structure after a post pass (using Ramer's method to chop curved segments).

Images
Figure 3.18

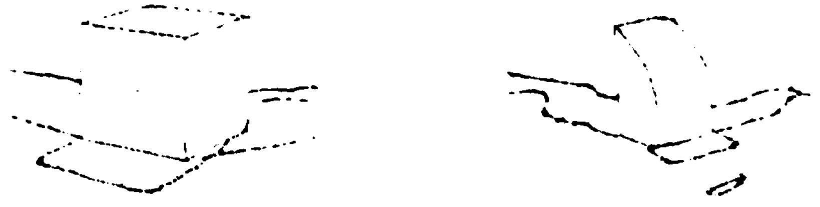


Edges For Figure 3.18
Figure 3.19

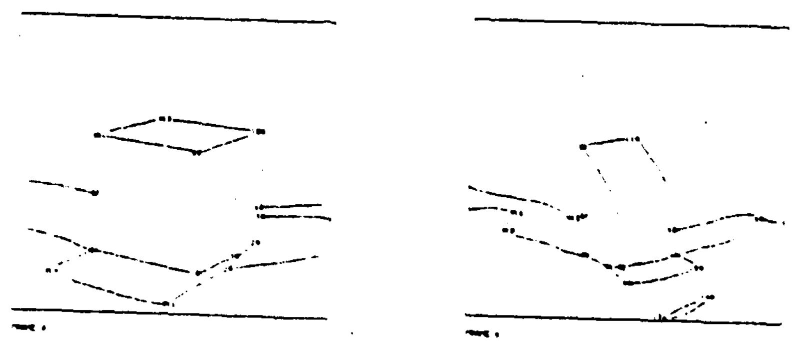


Vertex Marked Edges
Figure 3.20

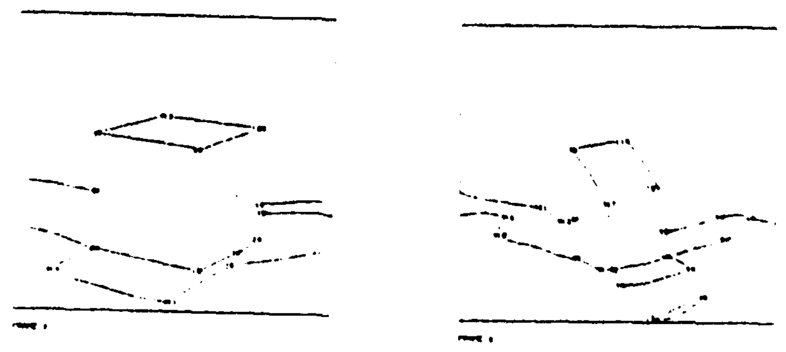
Vertex Initial (Dots) and Reconstructed (X)
Positions
Figure 3.21



Distance Edges Are Moved During Smoothing
Figure 3.22



V-S-S Graph Derived Plot
Figure 3.23



V-S-S Graph After Curved Segment Chopping
Figure 3.24

3.11. Notes on Additional Feature Dimensions

In most of the examples discussed gray scale intensity data has been processed. We, for the most part, restrict ourselves to this variety of data because it is the simplest to obtain. However, the processing we have so far described is not limited to simple intensity.

The most obvious form of multispectral data is color. We obtain color information by taking three pictures through red, green, and blue color filters. Many workers have proposed the combination of these spectra into hue and saturation. We choose not to do so because our approach is to extract critical discontinuities (edges) and organize them (form regions, contours, and vertices, each with property vectors). This can be done in the R-G-B space as well as in the hue-saturation one. In the R-G-B space, noise models are easier because they are exactly the same as for intensity alone (directly a result of digitization and sensor noise). If R-G-B is transformed to hue-saturation, noise functions are transformed and since this is a non-linear transformation, the noise transformation is non-linear. For this reason we choose, in general, to work with data in the form directly generated by sensors.

Any discontinuities, in any of the feature dimensions represents a significant point. Therefore, these discontinuities can be extracted independently and superimposed. In the set of all discontinuities, each individual one inherits properties from each feature dimension in which it occurs (i.e. the tacit assumption here is that if discontinuities co-occur in different feature spaces, but at nearly the same spatial position, they are all generated by the same underlying process). In this way, any system can not only be easily expanded to process color (i.e. light sampled in three frequency bands), but a wide variety of alternate feature domains also. Figures 3.25-3.26 show two color pictures, and the corresponding red, green, blue, and composite edge pictures.

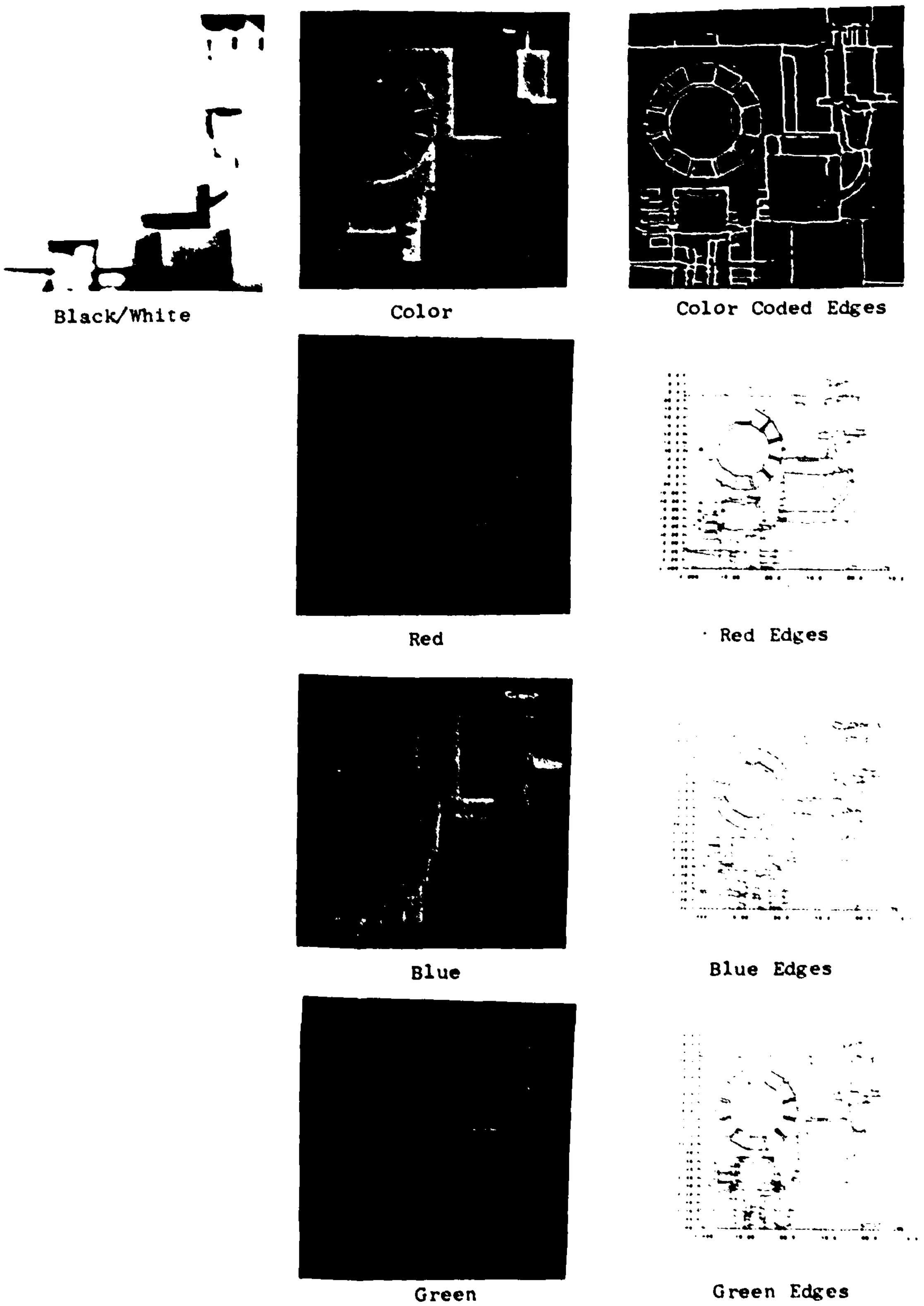
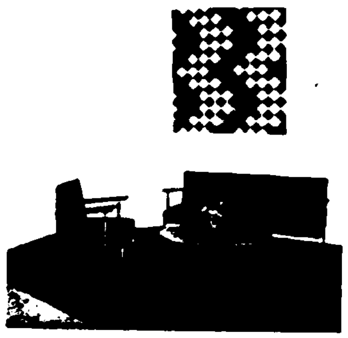
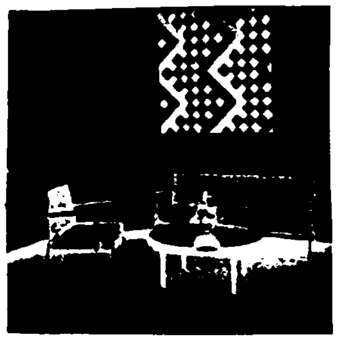
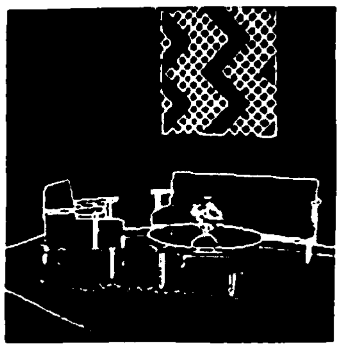
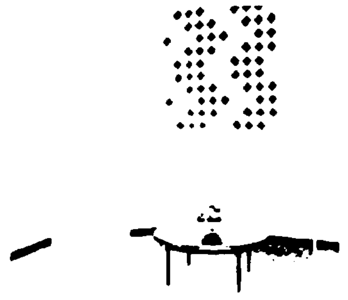


Figure 3.25. Color Edge Processing, Local Image

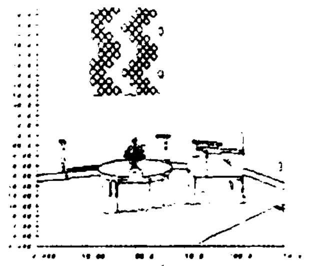

Color *


Color Negative

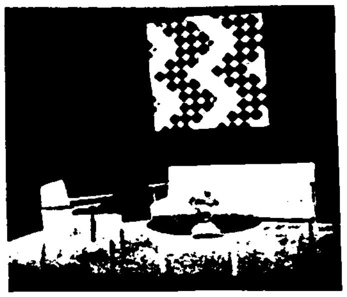

Color Coded Edges

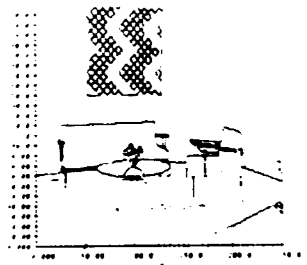

Black/White

Red


Red Edges **

This image is from [45], reduced from 800x400x8 bits to 252x238x9 bits.

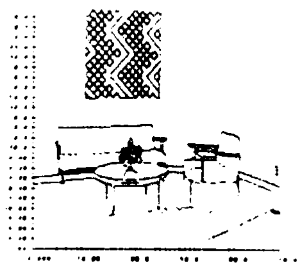

Blue


Blue Edges **

* Our images are negatives compared to those used in [45]. We process the color negative for this image.

** An unavoidable x axis reversal in the edge display.

Green


Green Edges **

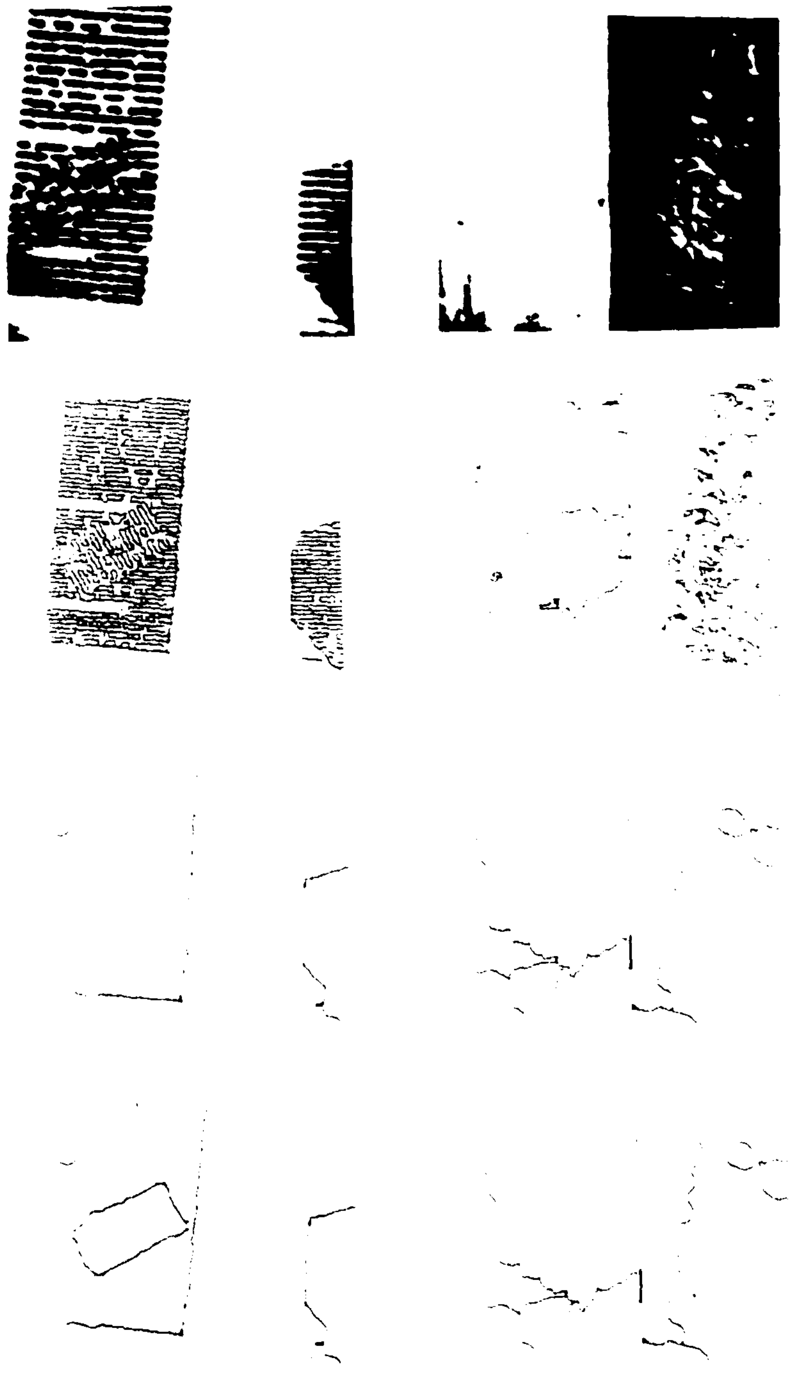
[illegible] lor Edge Processing. Image From [45]

3.12. Notes on Spot and Edge Textures

This system of region, vertex and string encoding does not directly process texture information. However, all regions which can not be processed as smooth are marked by the region processor as rough. These regions may be passed directly (roughness being a region feature vector component), or may be transformed to "smooth" by some texture smoothing process. We believe that "texture" is actually high-density shape information. Shape information with higher density than can be directly handled by the main line shape processes.

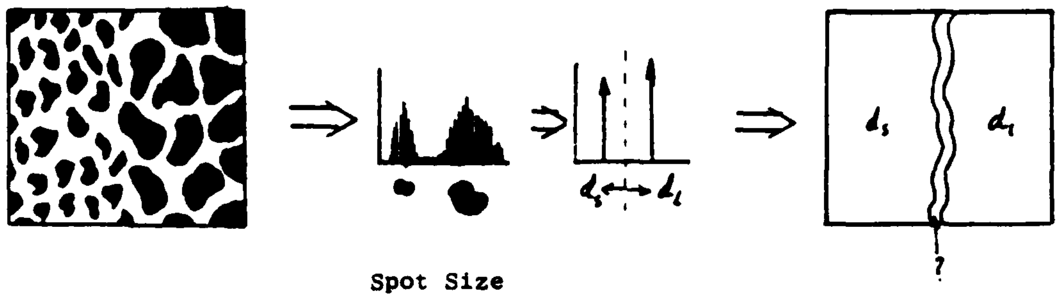
When a region has been too "busy" for the region process to handle, we use one of several primitive shape smoothers to form images which contain null areas for already smooth regions, smooth, non-null areas for regions having texture types compatible with the shape smoother, and "busy" areas for regions for which the shape smoother is in applicable.

These smoothed images may then be passed back through the edge detection system, for edges between the smoothed areas, and these edges precipitated into the global edge images (busy edges in smoothed regions are masked away). After all the known shape smoothing algorithms have been passed over rough regions, any still rough areas can be can be set to smooth and called "none-of-the-above" rough regions (Figure 3.27).

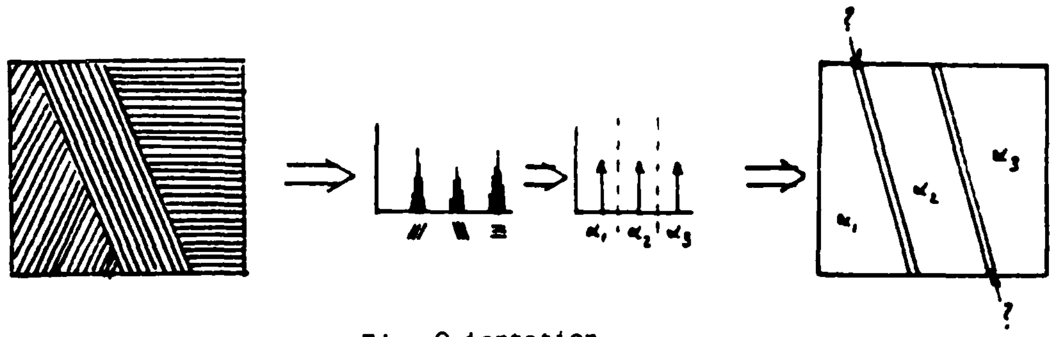

Textured Scenes
After Edge Detection
After Separating Smooth And Rough
After Separating Different Orientations

Texture Processing
Figure 3.27

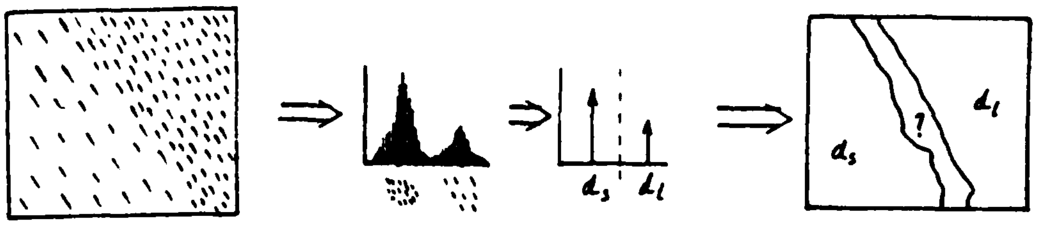
In this way, we are characterizing texture as microscopic shape information, in high enough density configurations to be characterized by statistical models rather than by specific graphical models. This approach allows much more integration between shape processing and texture processing. Each of our smoother processes may be thought of as a matcher for a given edge and edge connection histogram (Figure 3.23), as described by Marr. Marr proposed that such histogram characterizations of textures could be sufficient to mimic human texture discrimination.



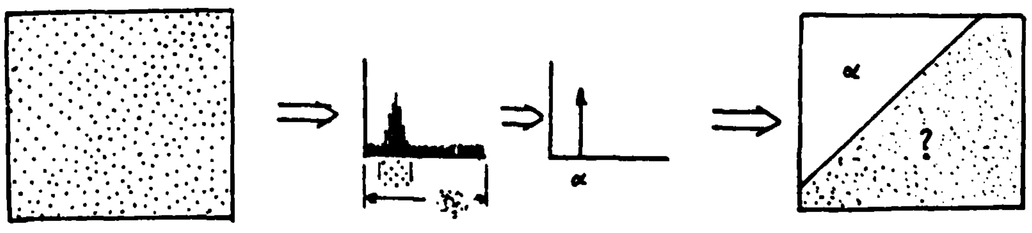
Spot Size



Edge Orientation



Edge Density



Constant Spacing vs.
Random Spacing

Example Texture Histograms
Figure 3.28

We claim that texture-like phenomenon can occur at any level of shape representation, and therefore cannot be considered by one simple formalism (like spatial spectral analysis). Also, texture processing can best be understood as high density shape detail, too dense to process explicitly. Therefore, texture can best be studied in the context of shape extraction and representation.

3.13. Conclusions

We have described a system which operates on single images to form complete vertex-string-surface graphs. Surfaces are aggregated prior to contour processing by using the idea of homogeneity as the basis for region segregation. This method allows edge closure and the generation of some classes of subjective contours.

The contour-vertex processing described finds vertices before contours, by examining local edge connect conditions (reflexive connections). This method allows contours to be labeled without path following (we use a pure scan line algorithm). Vertex reconstruction and contour smoothing is performed via a tangent line based technique. It is shown that detection of curvature discontinuities along contours is analogous to the detection of edges in image spaces.

Textured areas are segregated from smooth areas for optional additional processing. This consists of a variety of texture smoothing processes. Each of these, for a given class of textures, will produce smoothed images from rough ones. The various smoothed images may be edge detected and the results superimposed over previously detected smooth region boundaries. In the same way color images can be processed. The different color spectra are individually examined for edges with all detected edges being merged upon completion.

Chapter 4

Half Chunks, Depth, and Motion Correlation

4.1. Introduction

Detection of scene change has been performed via bulk correlation techniques which nominally reduce to picture subtraction or convolution, followed by segmentation and characterization of difference areas. In these approaches "difference" is examined very closely to the feature level (i.e. difference in intensities, or intensity statistics). Jain, Militzer, and Nagel [27] have used a method which compares local statistics to determine areas in scenes where change has occurred. Potter [49] has used image differences for object segmentation. Gennery and Burr have looked at bulk depth correlations, Gennery in outdoor scene contexts and Burr for machine parts. In this approach patches of one image are bulk correlated along a path of probable parralax in the matching image (a convolution operation). Marr does depth computation by matching symbolic entities from left image to right image, minimizing depth discontinuities [35]. We have approached the problem of change correlations in even a more symbolic manner. We look for similarities in the structure of abstract forms produced from stereoscopic motion picture sequences. This is similar to some of the modeling work of Baker, Baumgart, and Burr, except we do not restrict the variety of motion allowed for objects.

We acquire our structural information in the form of V-S-S graphs, one graph per input image. These graphs are re-encoded as half chunk graphs and compared for depth and motion cues. The first variety of graph is similar to that used by Underwood [62], however we encode more complete topological and geometrical information. The second kind of graph is entirely new. The graphs of several different images are matched, binding atoms and links of common types together. In a similar system, Dudani forms matches in vertex directed way [11]. Our matching scheme tends to be guided by points of high curvature or significant topological properties, but only because these have high discrimination value (tend to form unique matches).

This approach is desirable for several reasons. First, by matching on half chunk features, computational effort can be reduced by selecting pairs of atoms (or links) for association, consistent with uniquely matched "seed" atoms (links). Such a selection procedure assumes that some structural features (such as descriptive contour curvatures, vertices, or types of intensity variations), are so descriptive that they can only be matched to one corresponding feature from another graph, and that this matching is correct. This condition can be guaranteed by deferring image matching to a level at which each "component" to be matched is a significant part of a whole object.

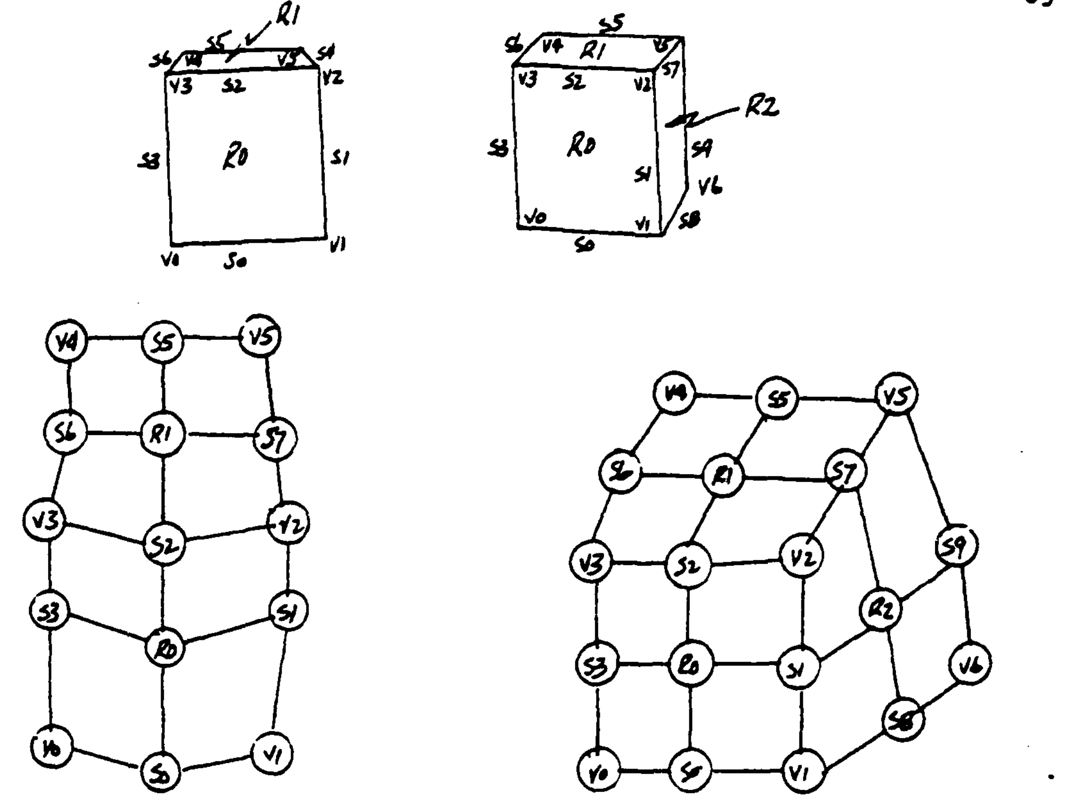
The second advantage we get from matching half chunks is that the various atom and link bindings which we obtain are easily interpreted in terms of rigid body transforms of object parts and eventually whole objects. Each aggregation of bindings around a given unique seed match gives rise to transforms from one subgraph to another directly.

Finally, our techniques are equally applicable to depth, motion, and image registration correlations. Depth correlations primarily differ from motion ones by the allowable transform taking one graph (the right one) to the other (the left one). Since the depth and motion transforms are easier than the registration ones, we have included them in our matching procedures.

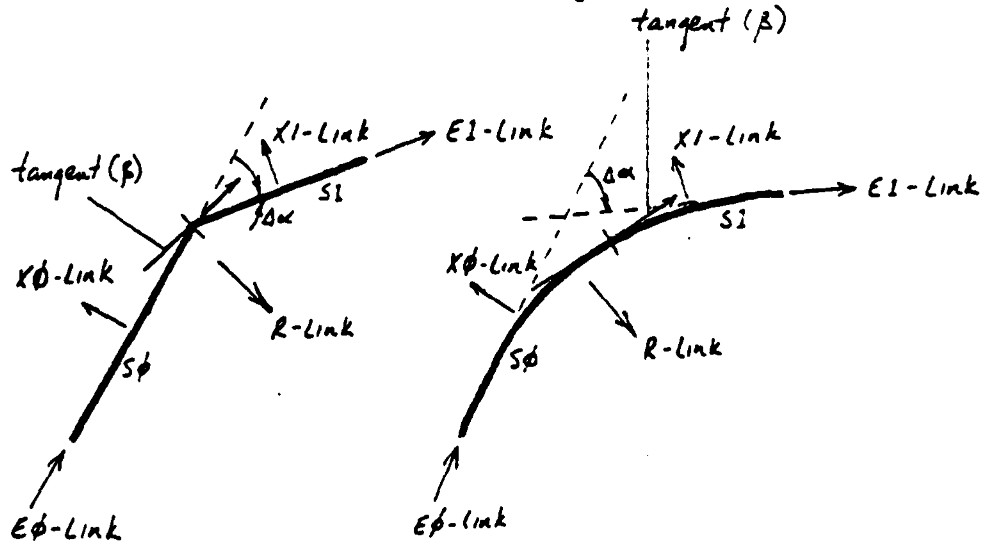
4.2. Graph Re-encoding (V-S-S to H-C)

In generating the single image graphs we found it useful to segregate the various components by their geometrical types (i.e. vertices, strings and regions). We found this convenient because each of the different types naturally surfaced from our data during different processing steps (small strings or edges first, regions next, long strings and tentative vertices after regions, and finally true vertices).

For the purposes of graph matching, we desired a uniform encoding for our graphs, in which all the information encoded into the V-S-S graphs could be used, but in more normalized form (Figure 4.1). To this end we have invented the "half chunk" (H-C) graph. Each "half chunk" represents a fragment of one half of a bounding contour for a region (or a length along a string). Geometrically it represents a single change in boundary curvature (or in the case of a vertex, a tangent angle discontinuity) at a point in space (3-space or 2-space), and a tangent line at that point. Topologically, it encodes five connection types, one across-region link (the R-link), two endpoint extension links (the E-links), and two across-boundary links (the X-links, Figure 4.2). The half chunk also carries a property list composed of features computed from the region and string feature vector lists.

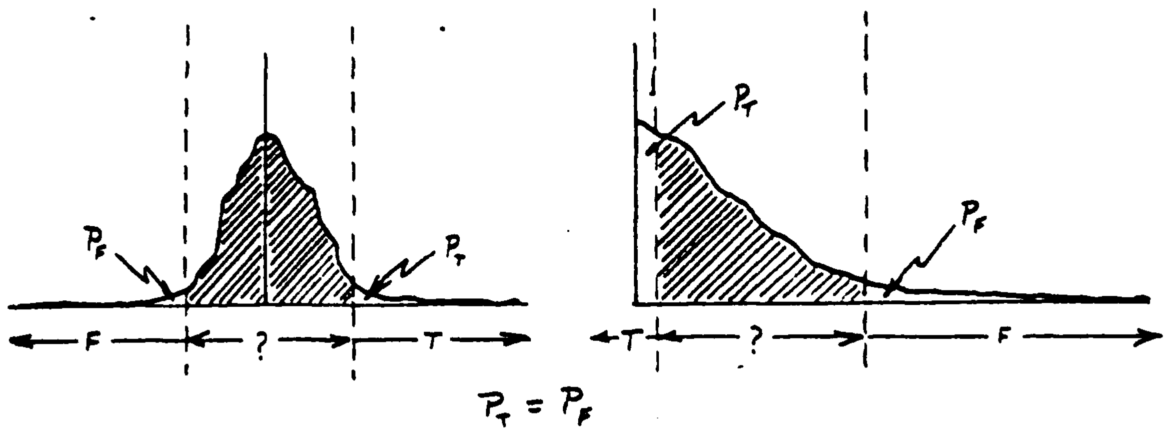


V-S-S Representation Of A Cube
Figure 4.1



Half Chunks
Figure 4.2

Each half chunk could carry a completely arbitrary floating point feature vector recording things like region color, intensity statistics, and contour diffusion (degree of blurring). However, for complex scenes this would be more than we require. We want to make and record decisions about the various feature dimensions, rather than record actual feature vectors. An array of decision features are computed from feature components encoded in V-S-S atoms. The noise statistics for these decision features are approximated, as modified normal distributions, and are used to trisect the feature spaces (Figure 4.3). True (t) records a strong detection of a feature, "maybe" (?) a weak detection of the feature, and false (f) a sure *non*-detection. This three state decision property is recorded rather than an actual feature vector component.



Two Sided Distributions One Sided Distributions

One-sided and Two-sided Decision Models
Figure 4.3

Each H-C decision feature is related to some question we would like answered concerning the intrinsic properties [3] of the image generating our graph structures. We summarize each decision feature as follows:

Feature 0-3; Intensity Chops

Intensity is scaled so that the minimum intensity is 0 and the maximum intensity is 1 (so as to be light level independent). Feature 0 is the region lightness minus the 0.5 level (for the region linked via the half chunk R-link). Is the region lighter than average or darker than average? Feature 2-3 are successive intensity chops (divides the half range, the quarter range, and the eighth range).

Feature 4; Boundary Diffusion

Nominal Diffusion is 3 (3 pixels) due to the imaging system. Is a specific boundary more diffused (i.e. likely to be a shadow)?

Feature 5; Lightest/Darkest

If the intensity is lower than 0.5, then this feature is the region intensity minus the darkest region intensity. If the intensity is higher than 0.5, then it is the lightest region intensity minus the region intensity. Regions which are nominally the lightest in a scene could be direct reflections of the light source (or a highlight). Darkest regions could be unlit (or deep shadows).

Feature 6; Intensity Step Size

This is the difference of intensities across a boundary. Is the boundary a light -> dark boundary or a dark -> light boundary?

Feature 7; Difference Of Boundary
Intensity To Region Center Intensity

This is obtained by subtracting the intensity along a boundary from the average region intensity. Surfaces which roll away from the observer get darker towards the boundary (less reflected light). Flat surfaces, with abrupt junctions to other surfaces have roughly constant lightness.

Feature 8; Variation Of Feature 5 Along A Boundary

This is the standard deviation of Feature 5 along the boundary. This detects a uniform lightness ratio across a surface-surface junction. This can occur when a surface is painted on another.

Feature 9; Intensity Variation Along A Boundary

This is the standard deviation from the mean intensity along the boundary. This feature becomes large along rolling surface boundaries.

Feature 10; Variation in Region Intensity

This feature is the standard deviation of average region lightness. If non-zero it implies the region is rolling and lit by a light source.

Feature 11; Average Roughness

This feature is the average edgedness for a region.

Feature 12; Boundary Real/Subjective Index

This feature is the ratio of real edges per edge for a bounding contour. It is a function of how broken the bounding contour was prior to region aggregation.

Feature 13; Depth Variation
After Depth Correlation Only

Planes are locally fitted to all correlated half chunks after a depth binding operation. Any H-C with an anomalous depth relative to its average plane is marked as having a depth variation. Highlights can appear as either under or over the average surface level. Occluded junctions may also display anomalous depths.

Feature 14; Motion Variation
After Motion Correlation Only

Planes are locally fitted to motion correlated half chunks. H-C's with anomalous velocities are marked. Shadows, highlights, and occluded boundaries can show this affect.

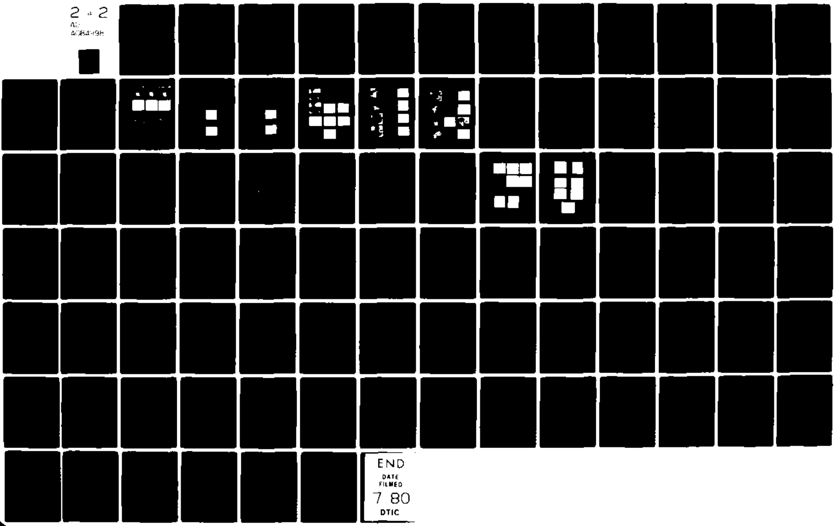
Feature 15; Region Embedding Level

This is not a decision feature like the rest. No statistical decision is involved. The (t) value represents the largest ground region. The (f) value represents the innermost level of embedding (highlights tend to be at this level). And the (?) value marks all intermediate region embedding levels.

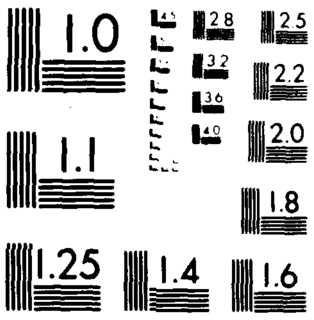
Using the half chunk we can encode regions, strings, and vertices in easy to traverse graph structures, which allow relatively easy graph matching (Figure 4.4). The modification or removal of relationships during a matching sequence (such as the disassociation of two adjacent regions), can be easily accomplished by simply breaking links. Insertion of new surfaces uncovered in the process of successive viewing can be easily incorporated by "ripping" boundaries in half and linking in the half chunks bounding the new uncovered surfaces (Figure 4.5).

AD-A084 998 ILLINOIS UNIV AT URBANA-CHAMPAIGN COORDINATED SCIENCE LAB F/G 5/8
VISUAL RECOGNITION OF ARTIFACTS BY COMPUTER.(U)
AUG 79 C J JACOBUS DAAB07-72-C-0259
UNCLASSIFIED R-851 NL

2 of 2
AD
A084998



END
DATE
FILMED
7 80
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

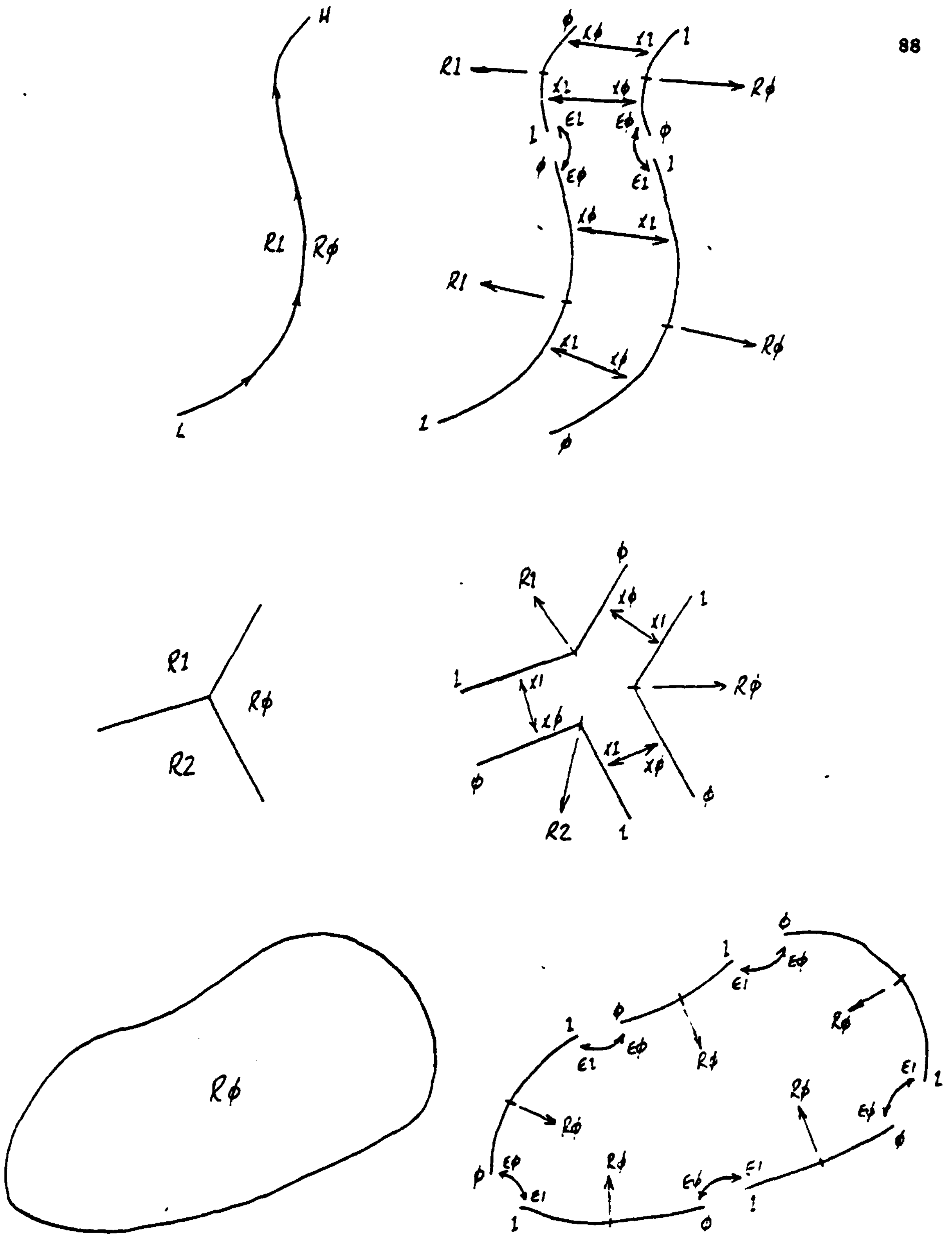
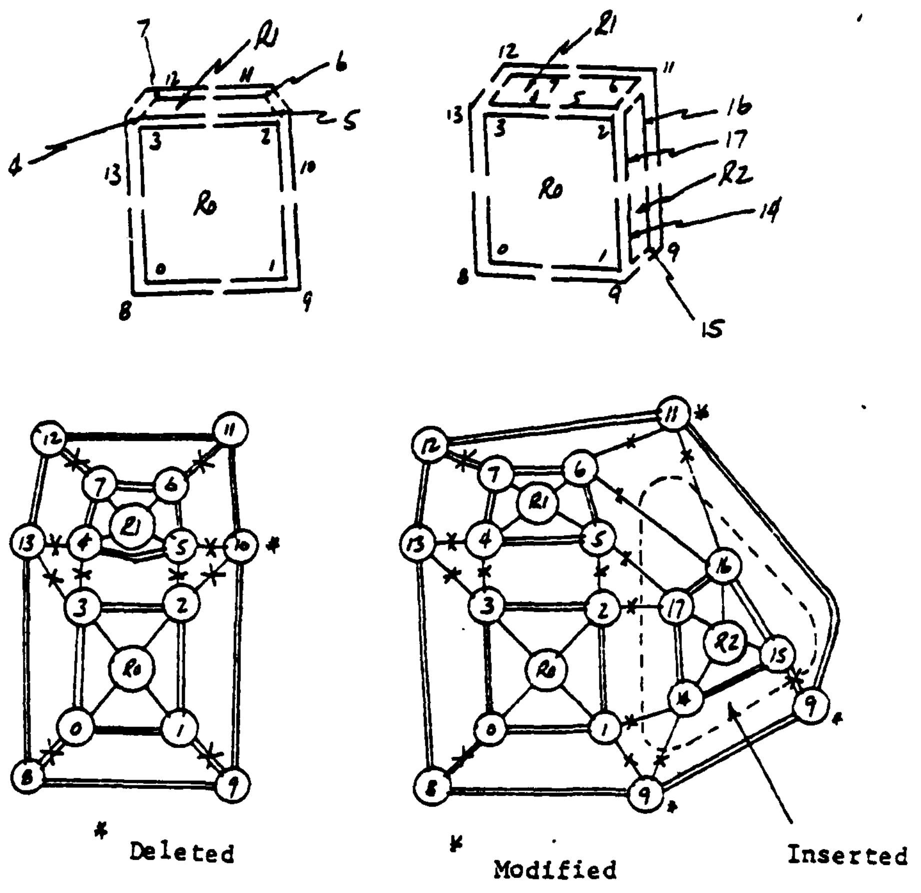


Figure 4.4. Encoding Regions, Strings, Vertices With H-C's



Region Insertion
Figure 4.5

Certain types of "interesting" structures become apparent in the H-C graphs. Vertices turn out to be circuits along X-links. Regions are formed by circuits along E-links (or by H-C's snaring the same R-link). A generalized "interesting" thing is something which makes chains and particularly closed chains.

4.3. Graph Matching (Depth and Motion Correlation)

Finding matches between non-identical graphs is in general a difficult problem, certainly much more difficult than finding simple isomorphisms. If we do not expect to find many actual isomorphic matches, then a natural way of ordering all possible non-isomorphic matchings is via the idea of cost [55]. We can invent some basic atom-atom and link-link cost functions, and can define an overall match cost as some function of the individual component (atom or link) costs for all the bindings between the matched graphs. If we have good component cost criteria, then good graph matches will also have low cost. We, therefore, need to search the possible graph binding space (this space consists of all possible atom-atom, and link-link bindings between two input graphs), for minimum cost component binding configurations.

If we choose a metric (such as Euclidean distance) as the basic element-element cost function, we may need to evaluate large numbers of bindings to find the optimal whole graph binding. This computational expense can be significantly reduced by typing the individual atoms and links and allowing comparisons only between equivalent types. We can, in many cases, do this typing without any real loss in power. We would not generally compare surfaces with vertices, atoms of any type with links, or contours considered very diffused with ones razor sharp.

This brings us to our particular match scheme. Three critical ideas have been useful in patterning our algorithm. First, we constrain any final matching of graphs to those in which only one-to-one atom (link) and one-to-none atom (link) matches occur. Second, we match primarily by "covering", and only secondarily by atom (link) to atom (link) metrical cost. And finally, matching processes are seeded at all highly descriptive unique matching points, and allowed to compete with each other in a parallel fashion for the remaining unbound atoms and links. In good situations, most of the individual matching processes eventually grow into contact with their neighbors, find compatibility between the neighbors and themselves, and merge with the neighbors. In this way large areas of a graph structure are matched in parallel.

Generally, we are require six costs in a graph matching scheme:

A is an atom from graph Ga
B is an atom from graph Gb
La is a link from Ga
Lb is a link from Gb

4.3.1.	$f(A,B)$	Cost for matching A and B
4.3.2.	$g(A)$	Cost for matching A to null
4.3.3.	$u(B)$	Cost for matching B to null
4.3.4.	$h(La,Lb)$	Cost for matching La to Lb
4.3.5.	$k(La)$	Cost for matching La to null
4.3.6.	$v(Lb)$	Cost for matching Lb to null

For the cost of an entire match, we select a cost (1-6) for each atom in Ga (for all A) and Gb (for all B). We sum the selected costs. We wish to find the matches where the cost is minimal.

4.3.7. $$C = \sum f(A_i,B_j) + \sum g(A_i) + \sum u(B_j) + \sum h(La_i,Lb_i) + \sum k(La_i) + \sum v(Lb_j)$$

In the most general atom matching directed scheme, we will necessarily have to check every atom A (in graph Ga) against every atom B (in graph Gb). And in the case where the costs, particularly the atom match associated costs (1-3), vary as function of things already matched, even more comparisons may be required. The link associated costs (4-6) will be designed to grow primarily as a function of mismatches which are a function of badly matched atoms, rather than mismatches due to atoms matched to null. This complicates the general scheme by predicating the link match cost on the atom matches. Finally, if we have atoms of different types and links, it is not immediately obvious how we should weight the components of graph matching cost associated with each.

We get around most of these problems by formulating a simple "covering"-based costing scheme. If we type atoms, we need never compare atoms of different type. They always are known to yield infinite cost. We may use the array of three state properties in the half chunk in the same manner as a type if we broaden the idea of "being of equal types", to the idea of "being potentially of equal types". That is, atom A covers atom B if A could be B and B could be A. In our three state scheme this kind of check is easy. The t (true) property covers ? (maybe) and t, the f (false) property covers ? and f, and the ? property covers all (Figure 4.6). It may seem that simple three state feature properties would be very weak, and indeed individually they are. However, the three values are sufficient to answer, any Boolean question posed concerning a given grouping of feature vectors, and if many such questions are asked, a powerful discriminator can be formed from the answers.

Covering Rules:

$t \oplus t = t$

$f \oplus f = f$

$f \oplus t = ?$

$a \oplus b = b \oplus a$

$a \oplus ? = a$

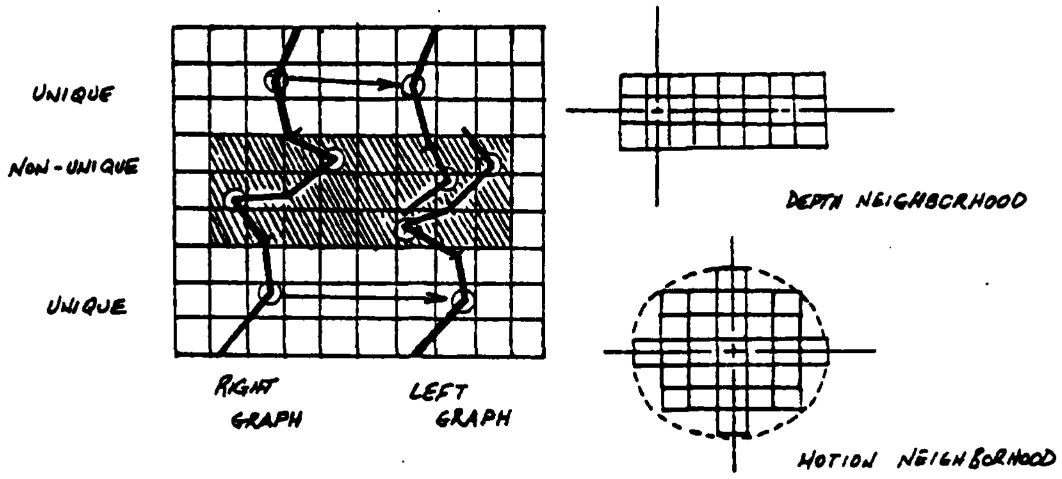
Example:

$$\begin{array}{cccccc l} t & t & t & f & f & f & \\ ? & ? & t & f & ? & ? & \\ \hline t & t & t & f & f & f & \text{covers} \end{array}$$

$$\begin{array}{cccccc l} t & t & t & f & f & f & \\ ? & t & f & f & t & ? & \\ \hline t & t & ? & f & ? & f & \text{does not cover} \end{array}$$

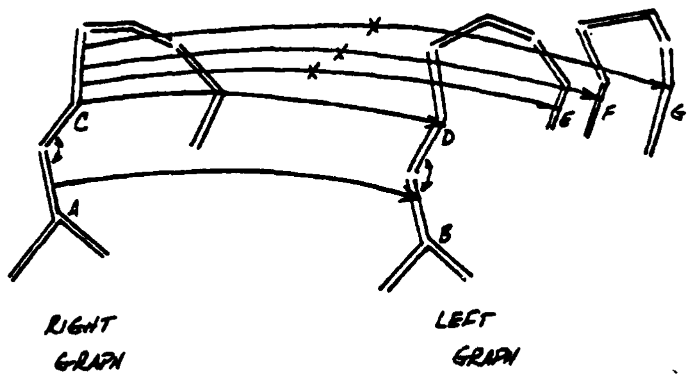
Covering Rules
Figure 4.6

For two half chunk atoms to be possible matches, they must "cover" each other, be geometrically similar (i.e. close enough in tangent, curvature, and size), and must be involved in encoding the same topological relations (i.e. must have compatible link structure and be either all vertex derived or contour derived). The matching algorithm uses one graph as a template for the other. About each H-C in the template a neighborhood (circular for motion matching, along the direction of parallax for depth matching) is expanded and all H-C's from the other graph falling in the neighborhood are examined for covering, topological consistency and metrical cost. Any H-C's with unique (or no) matches are marked (bound to the unique match, or marked as matching null). Each unique match acts as a seed around which a local interpretation may grow (if no unique matches are generated the matching with the lowest metrical cost is chosen as a seed, but this has actually never occurred except while processing some test datasets - Figure 4.7).



Neighborhood Expansion, Seed Formation
Figure 4.7

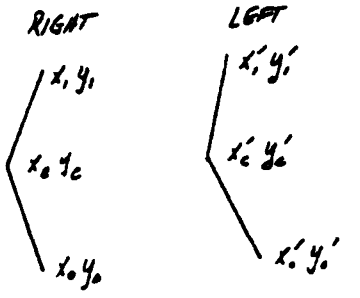
After binding up all unique matches, some previously ambiguous matches (H-C's matching more than one in the other graph), will now be unambiguous. This happens because of link consistency. If atom A matches uniquely atom B, A connects through an endpoint link to C, B connects through an endpoint link to D, and C ambiguously matches D, E, F, and G, then after binding A and B, clearly we should bind C and D (Figure 4.8). We can therefore imagine unique bindings causing new unique bindings through any links. By a relaxation scheme, these influences are propagated until the various seed centers grow to collision.



A Unique Influencing Adjacent Non-unique
Figure 4.8

When two seed centers collide, if they represent mutually consistent bindings (i.e. they both cause the template atoms at the collision area to be bound to the same atoms in the other graph) seed growth stops, and the matching is locally finished. If the seeds disagree about particular bindings, the interpretations with the lowest metrical cost are adopted. Thus, one seed may steal atoms and links from another, and may ultimately obliterate the other. Because of our covering-based method for seed selection, we generally get good seeds evenly spread over the image graphs. Therefore very little competition actually occurs between seeds, and rarely is a seed totally absorbed by its neighbors.

Just as the neighborhoods expanded for motion matching differ from those expanded for depth matching, the metrical cost functions differ (Figure 4.9). For depth we allow small movement in the parallax for no cost, but require close fit in tangent, curvature and non-parallax variation. For motion we allow small movements in X or Y, but constrain tangent (though not as much as in the depth case), and curvature.



Motion Cost:

$$c = \max\left(\sqrt{(x_c - x_c')^2 + (y_c - y_c')^2},\ \sqrt{(x_1 - x_1')^2 + (y_1 - y_1')^2},\ \sqrt{(x_0 - x_0')^2 + (y_0 - y_0')^2}\right)$$

Depth Cost:

$$c = \max\left(|y_c - y_c'|, |y_1 - y_1'|, |y_0 - y_0'|\right)$$

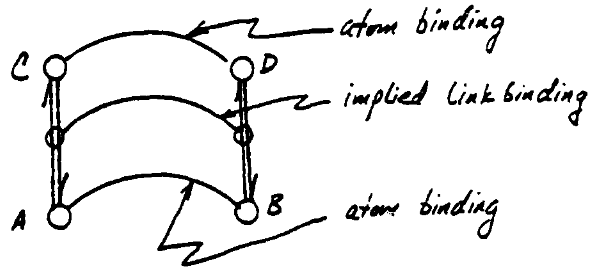
(no penalty for parallax shifts)

Depth and Motion Costs
Figure 4.9

For both cases, once a binding transform for one H-C is computed (from a unique binding), this transform is used for computing cost for connected H-C's. It is assumed that objects are relatively rigid.

4.4. Interpreting Bindings (Transforms and Graph Merger)

After a completed matching sequence we have the two original graphs (i.e. their atoms and links), and a set of atomic bindings (which imply link bindings - if A binds to B, A is linked to C, C binds to D, and B is linked to D, then the link from A to C is bound to the link from B to D - Figure 4.10). We must now use this binding information to fuse the two graphs, forming new graph elements associated with 3-space coordinates vectors (in the case of depth matching), or velocity vectors (in the case of motion matching).



Link Matching Implied By Atom Matches
Figure 4.10

The actual computation of a depth or a velocity for single half chunks, or clouds of half chunks in connected subgraphs is really quite straightforward matrix algebra (on potentially overspecified systems). We have chosen to simplify our depth computations by assuming that the objects we see are relatively far from the camera system, minimizing perspective distortions. If this is true, Z (depth) becomes negatively proportional to parallax.

$$p\ (\text{parallax}) = x_{right} - x_{left}$$

$$\left.\begin{aligned} x &= x_{right} \\ y &= y_{right} \\ z &= C(k-p) \quad \text{choose } k \,\&\, C \text{ during calibration} \end{aligned}\right\} \text{Simplified Depth Computation}$$

The main advantages of this approach to computing X, Y, Z are simplicity, and the inherent isolation of parallax errors to only the Z component. X and Y measurements are relatively accurate (good to one pixel position), and P (parallax) measurements are good for computing relative depth. However, in translating X, Y, P into absolute X, Y, Z significant errors due to P can occur. It is better to isolate these purely in the Z coordinate.

For motion transform computation we assume small overall motions. Therefore we ignore any overall scale changes. We compute three different motion transforms, one assuming simple translation, one assuming translation and a rotation in the plane of the image only, and one assuming translation and a general rotation in 3-space. We select the most complicated transform which is not singular (or near singular).

Translation Only Transform:

$$X' = X + T \qquad X = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad X' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \quad T = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

$$T = \frac{1}{n} \sum_{i=1,n} (X_i' - X_i)$$

(only one corresponding point required)

Translation and 2-d Rotation in the Image Plane

$$X' = RX + T \qquad R = \begin{bmatrix} a & -b & \phi \\ b & a & \phi \\ \phi & \phi & 1 \end{bmatrix}$$

$$t_3 = \frac{1}{n} \sum_{i=1,n} (z_i' - z_i)$$

$$\begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2 \\ \vdots \\ x_n' \\ y_n' \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & \phi \\ y_1 & x_1 & \phi & 1 \\ x_2 & y_2 & 1 & \phi \\ y_2 & x_2 & \phi & 1 \\ & \vdots & & \\ x_n & y_n & 1 & \phi \\ y_n & x_n & \phi & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ t_1 \\ t_2 \end{bmatrix}$$

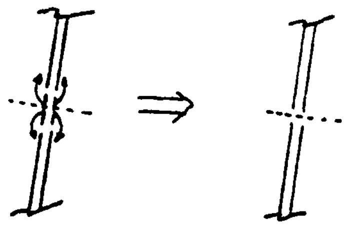
Translation and 3-d Rotation

$$X' = RX + T \qquad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad T = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

$$\begin{bmatrix} x_1' \\ y_1' \\ z_1' \\ x_2' \\ y_2' \\ z_2' \\ \vdots \\ x_n' \\ y_n' \\ z_n' \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & z_1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & z_1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 0 & 0 & 1 \\ x_2 & y_2 & z_2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & z_2 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 0 & 0 & 1 \\ & & & & & & \vdots & & & & & \\ x_n & y_n & z_n & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & z_n & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_n & y_n & z_n & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} \\ r_{12} \\ r_{13} \\ r_{21} \\ r_{22} \\ r_{23} \\ r_{31} \\ r_{32} \\ r_{33} \\ t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

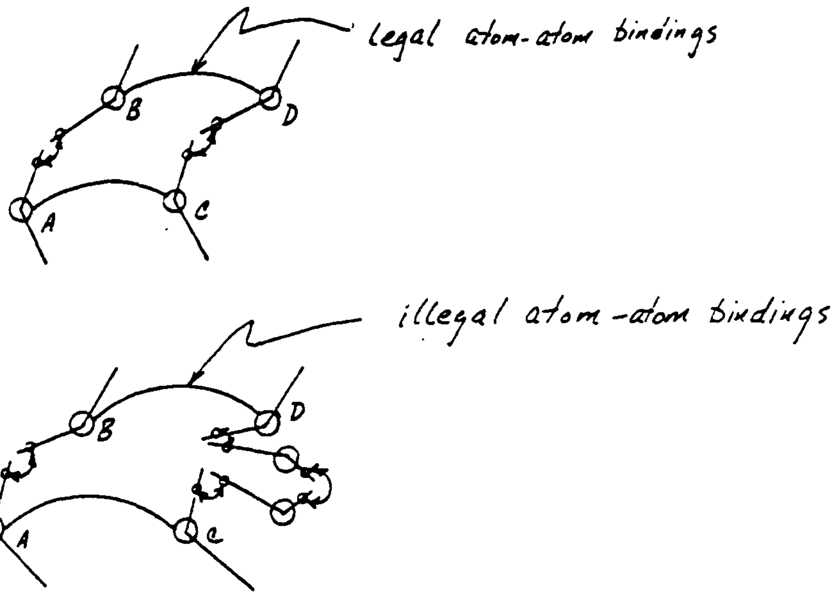
As has been shown, computation of transforms of various kinds is easy following the graph match. What is actually much harder is to produce an new graph (3-dimensional, or with moving half chunks) which is consistent and preserves as much of the associations of the original input graphs as possible. Producing this merged graph is a heuristic process, however some guiding principles are apparent.

If the basic signal data encoded in our graphs is good (i.e. accurately reflects object structure), we expect nearly perfect matching between individual scene graphs, where the underlying object structure is common. In fact, we get this criterion as long as we correct for one problem. We need to allow for boundary expansion or contraction due to slight changes in object viewing perspective. We deal with the problem by allowing small, unbound half chunks surrounded by larger, bound half chunks to be absorbed into their bound neighbors (Figure 4.11).

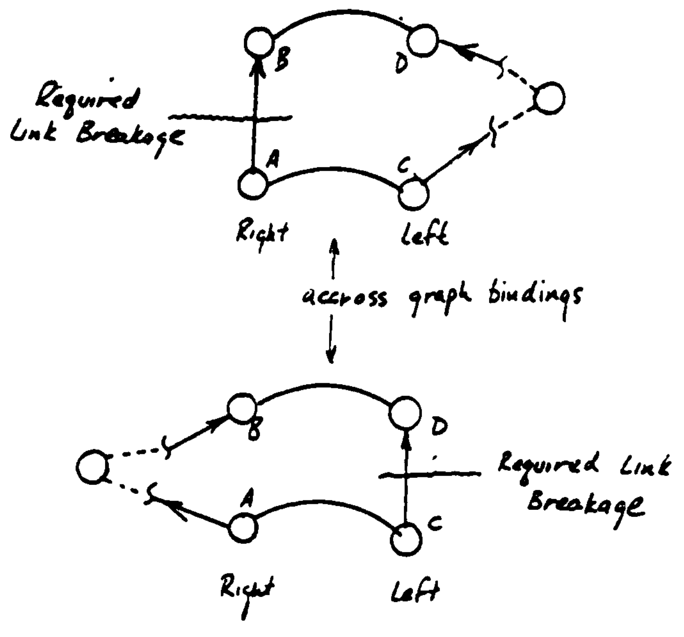


Boundary Contraction
Figure 4.11

The basic strategy for graph merging is based on believing atom-atom bindings, and breaking in-graph links until each graph may be unambiguously overlayed over the other. This basically means that atoms in one graph may have links (in one of the five possible terminals) where atoms in the other do not, or corresponding atoms may both be missing corresponding links, but if both have links they must be compatible. That is, if endpoint A links to B, and endpoint C links to D, and A binds across graphs to C, then B must bind to D. If this is not true then one or both of the in-graph links (A to B, or C to D) must be broken (Figure 4.12). The link which is broken is the one which can be explained most easily as an artifact of newly uncovered data, seen in one image (an therefore its graph), but not seen in the other (Figure 4.13).

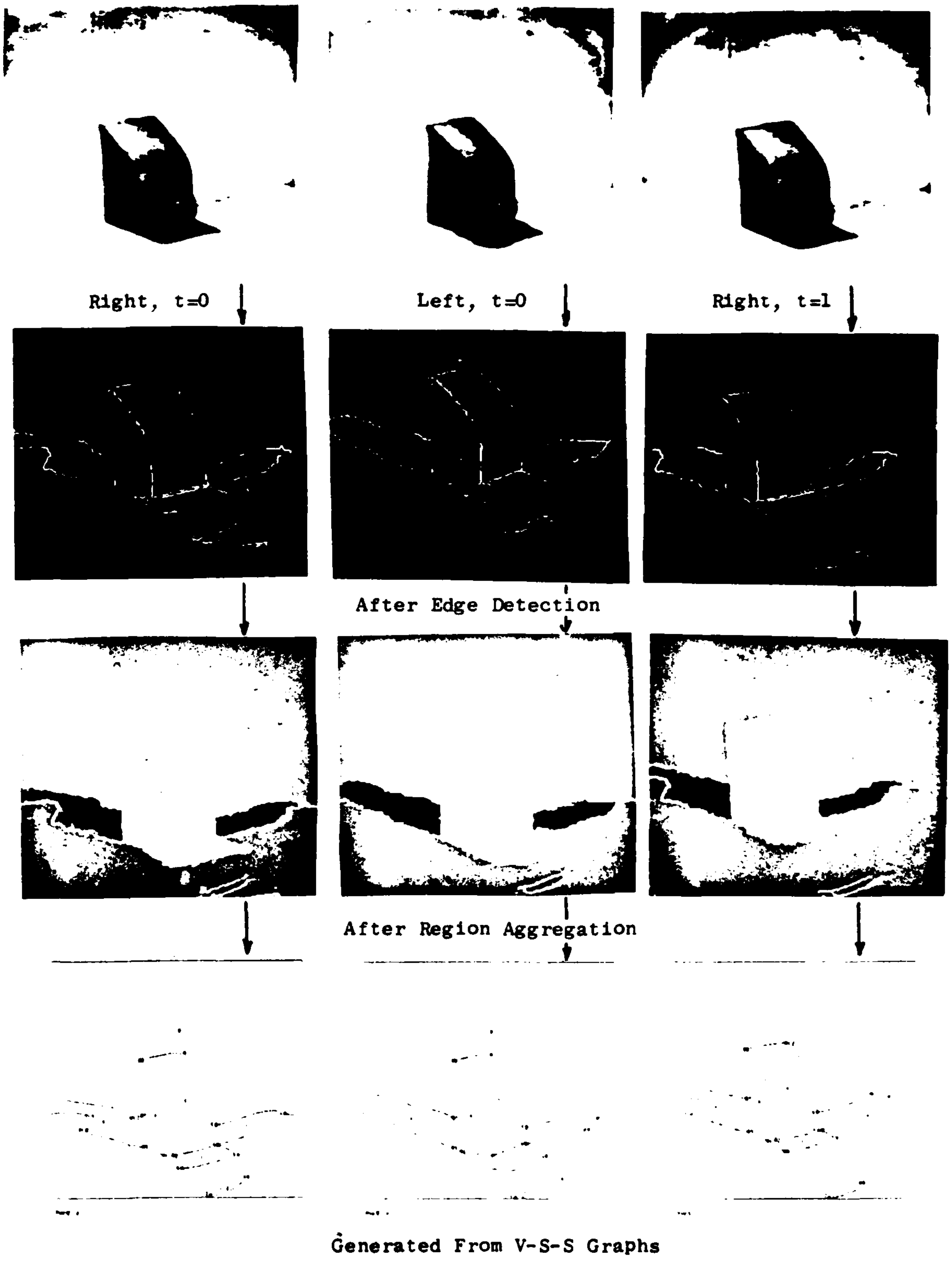


Legal and Illegal Bindings
Figure 4.12

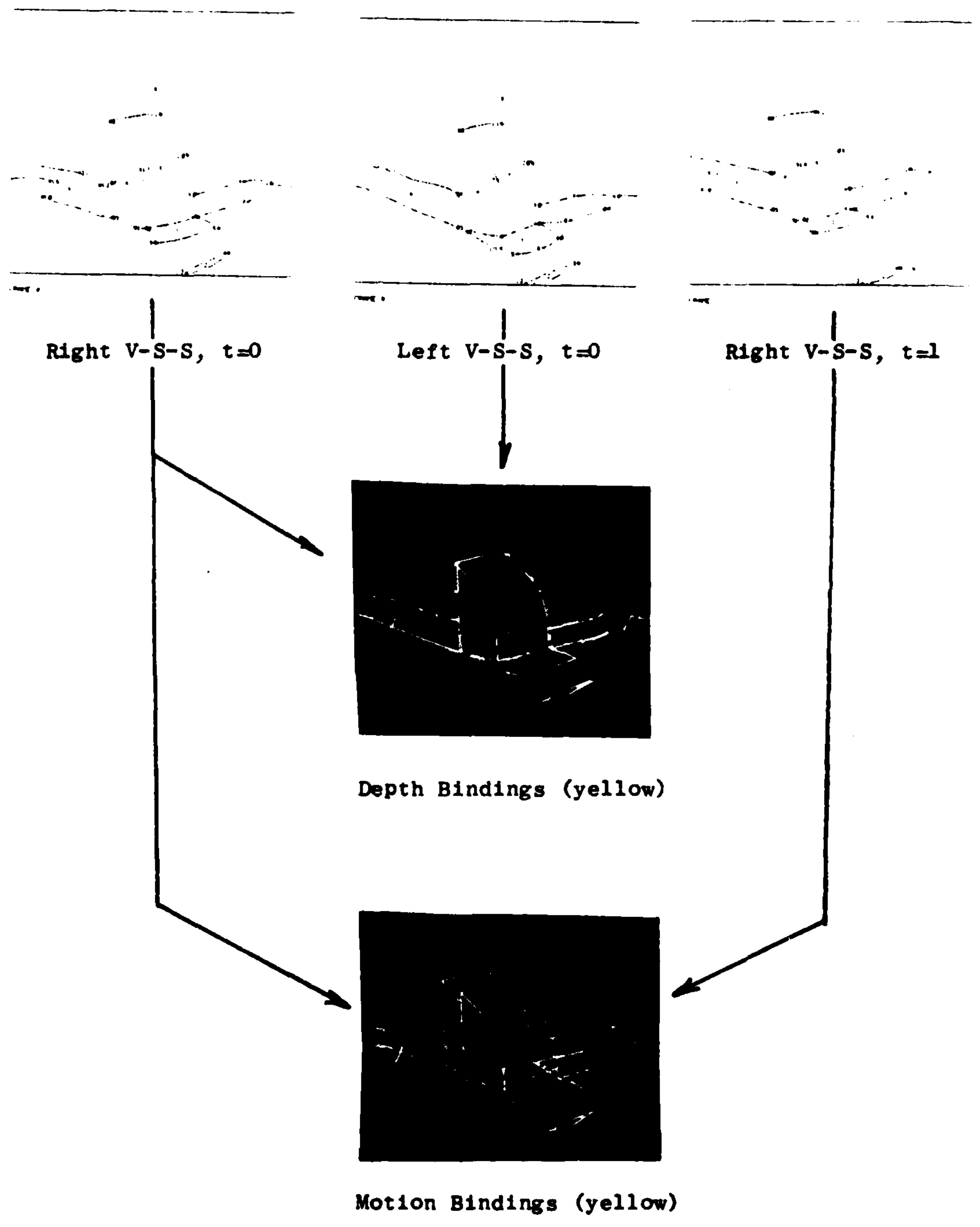


Links Broken Because of Inconsistency
Figure 4.13

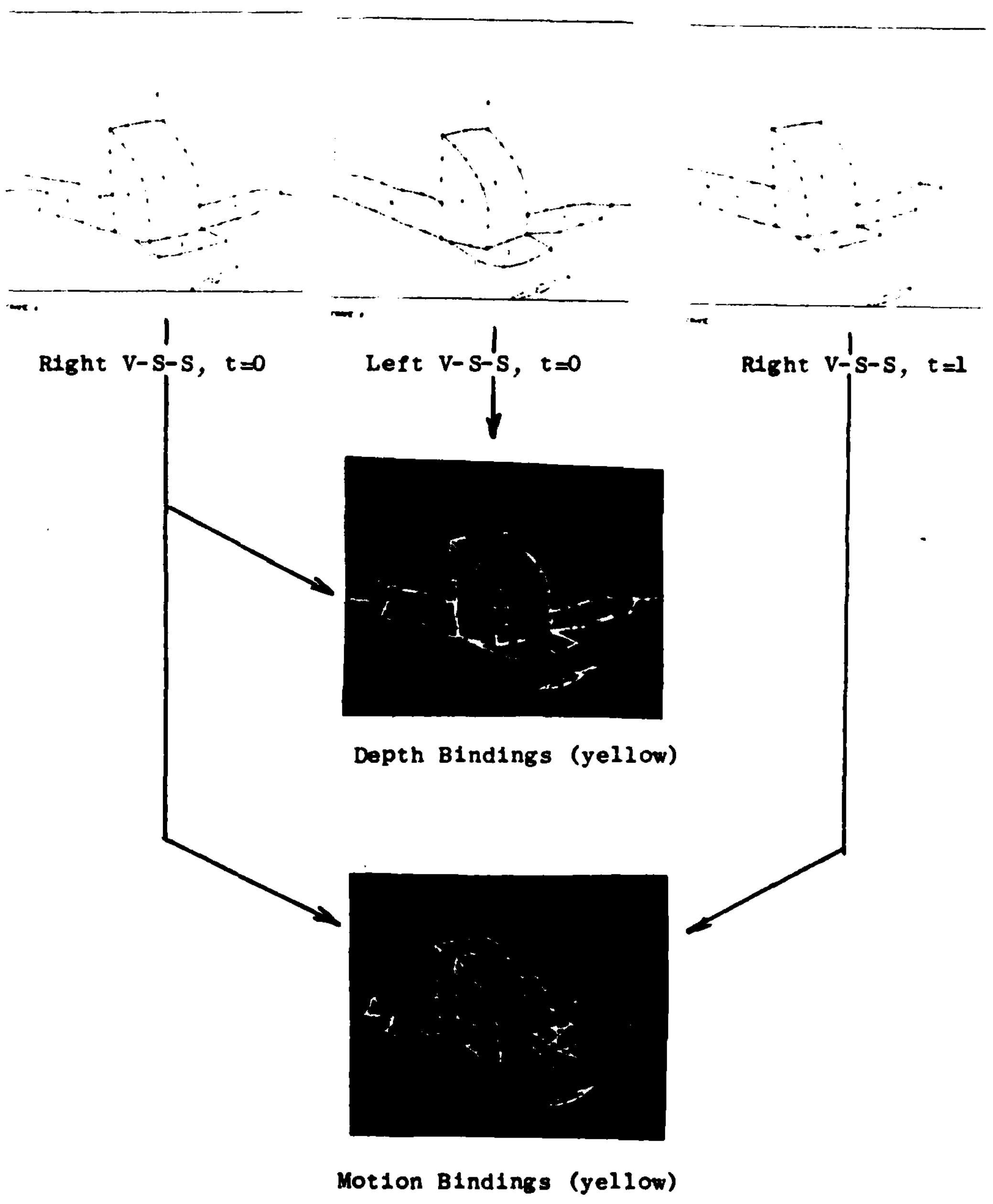
Figure 4.14 shows three images processed up to the basic V-S-S graph form (A. right t=0 image, left t=0 image, right t=1 image; B. Edges; C. Image constructed from the region assignment data; D. Tracing constructed from the basic V-S-S graphs). Figure 4.15 shows the basic V-S-S form broken in an object segment directed way (by segment curvatures - Ramer's method), converted to half chunks and correlated for depth and motion (bindings are marked in yellow). Figure 4.16 shows the V-S-S form broken into smaller pieces, converted to half chunks, and correlated. Figure 4.17 shows the same results (in less detail) for an interesting metal cutout (the 3-d model is rotated in three dimensions for display). Figures 4.18 and 4.19 show other scenes processed for object models (only one view and one time interval is displayed - objects found will be displayed in the next chapter).



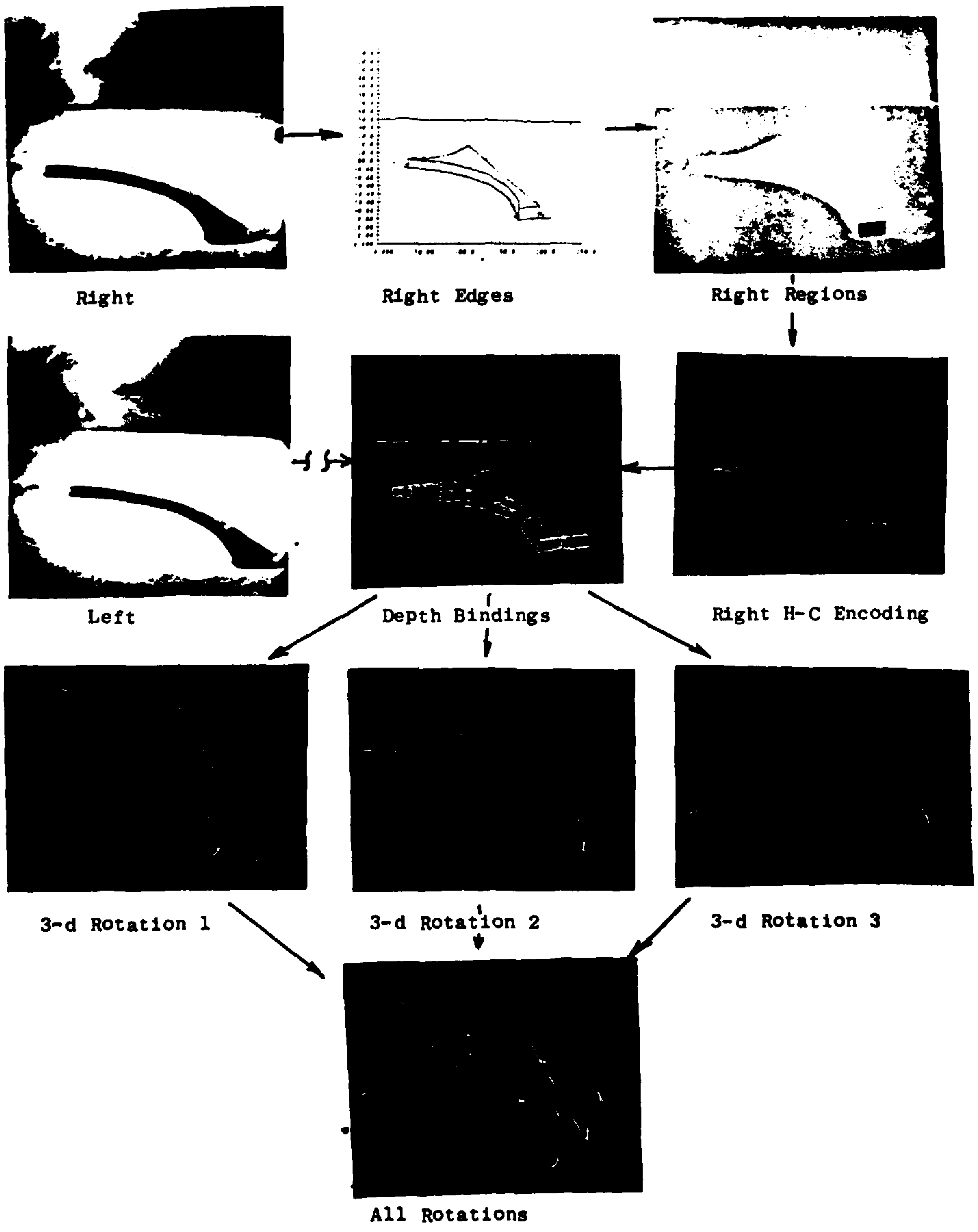
Processing Prior to Matching for Depth
and Motion
Figure 4.14



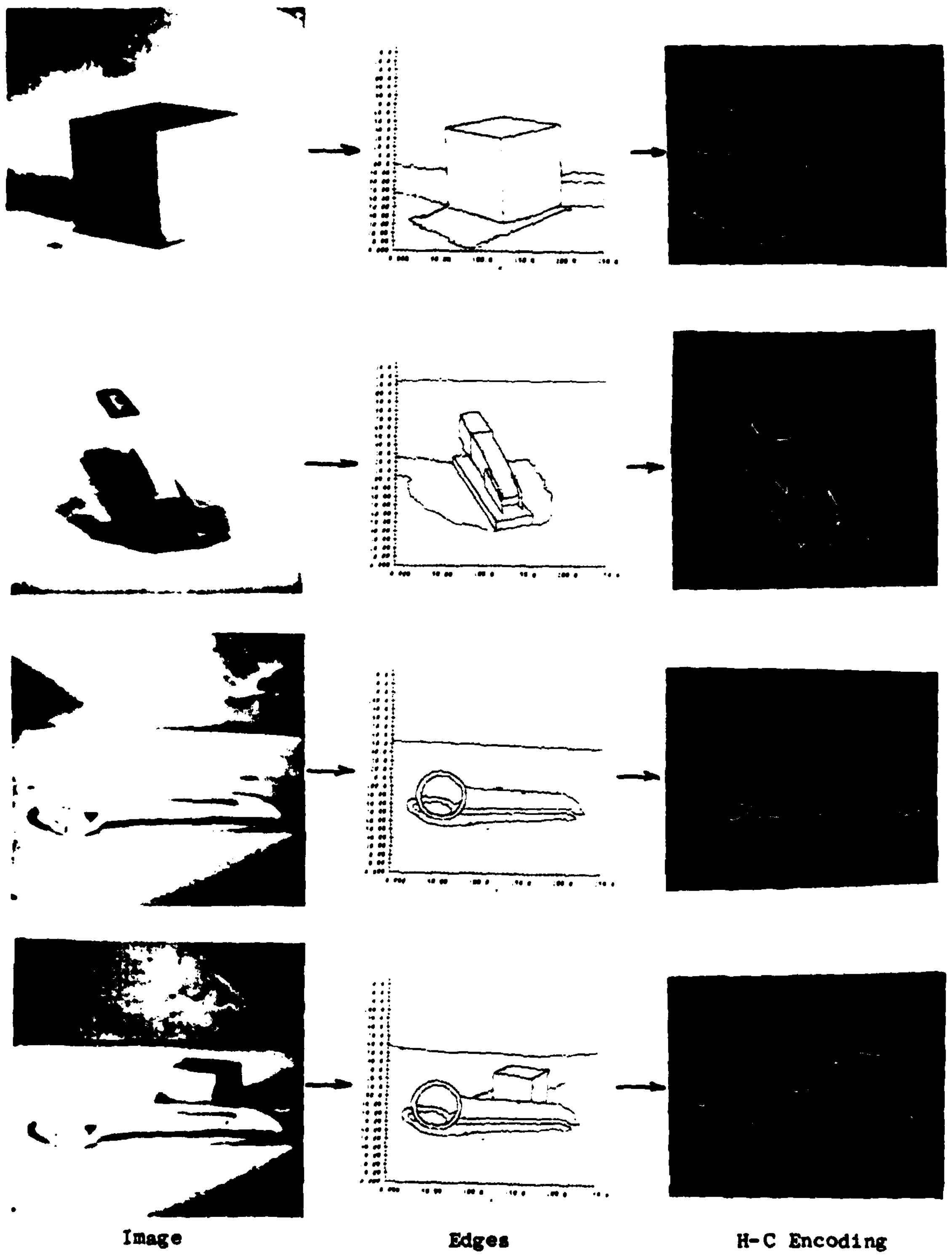
Depth and Motion Bindings For Object Directed Segments
Figure 4.15



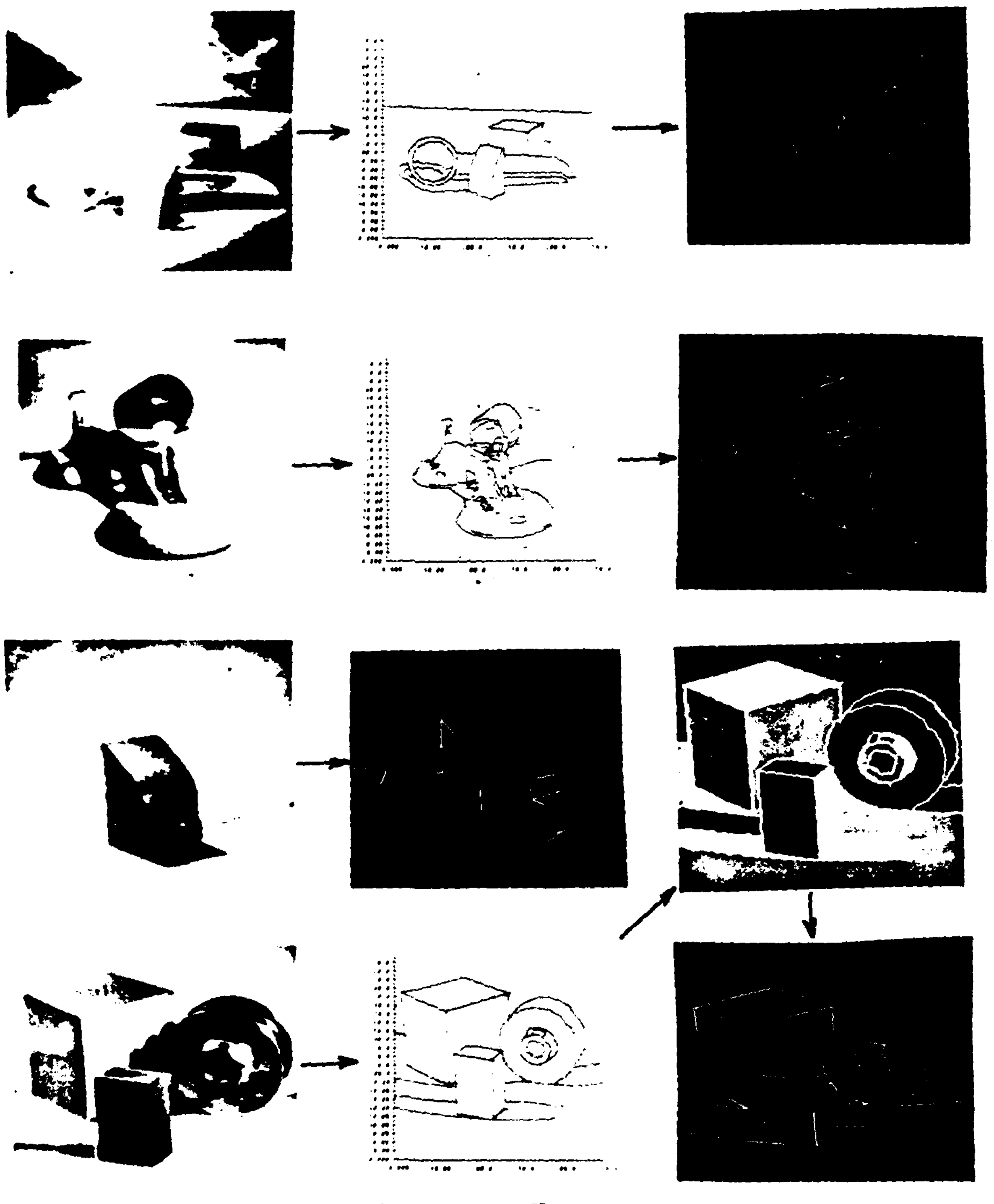
Depth and Motion Bindings For Small Segmets
Figure 4.16



Depth Bindings and Model Building .
Figure 4.17



More Objects and Object Clusters
Figure 4.18



Even More Objects and Clusters
Figure 4.19

4.5. Conclusions

We described a method for correlating visual information at the level of object structure, via a graph-based formalism. This method is very good for wide angle depth correlations, motion correlations and correlations where the transformation taking one graph to another is ill-defined. This method is powerful when used in conjunction with graphs made up of individually powerful (in terms of discrimination power) atomic pieces of object structure. We have introduced such an atomic unit of structure for the world of man-made artifacts (or objects primarily recognized by their shape), in the half chunk. We should note that the idea of the half chunk does not depend on graph matching, nor graph matching on the half chunk (we could have graph matched on the V-S-S graphs, but with more complexity due to the different atomic types, vertices-strings-surfaces).

Two major problems which arise when employing our graph oriented approach, are the problem of getting good graphs, and the problem of designing good atom-atom correlates. The latter problem we have addressed by characterizing atom (H-C) properties as three state decisions properties and using a "covering" based comparison scheme. The former problem is unsolved, and requires significant technical effort. Our results in the area are promising enough to make us think obtaining good graphs is possible.

Chapter 5

Object Identification

5.1. Introduction

Object models have traditionally been represented as either semantic net like graphs [65][62][59], or as object centered, rigid body, surface descriptions. The semantic net approach is deficient because dimensional information needed for exact object reconstruction is not incorporated. The surface description form has been used for 3-d hidden line/surface graphics, and some recognition work. Baumgart's system was primarily a graphics based system, which was applied to object representation. Baker tracked contour information, but still produced solid models which represent surface samples. Perkins stores 2-d models but also uses an object centered representation. Burr represented objects as object centered 3-d segments. Object centered models are good for display, however, are deficient for recognition because each new viewing angle generates models which are parametrized in different spaces. To match models, first, correspondences must heuristically be established, and at least one model reparametrized (so both are in a common coordinate system). This extra processing obscures the matching process. It has been our conclusion that object representation and subsequent recognition does not match either of the two previously described modeling techniques well.

Several works have suggested modeling techniques which instantiate some critical object feature relations. The generalized cylinder approach is like this. Many objects can be captured by specifying relations along major axes [44][37]. The trouble is that every object variety does not have these major symmetry axes.

Let us first list the requirements of a model representation scheme and a method for employing such a scheme for recognition:

1) The modeling method should potentially be able to capture object structure exactly.

2) The modeling method should be able to capture partial object descriptions (in the sense of missing views).

3) The method should allow for representation of objects at varying levels of exactness.

4) Models should be built by the system automatically. It should be possible to enhance a model with new information to form a more exact model.

5) Models of varying exactness and completeness should be comparable.

6) Modeling should take into account redundancy of structural components in real objects to form more compact representations.

7) It should be possible to recognize two very different objects as different with less computational effort than to recognize two similar objects as different.

8) Matching should not require exact matches.

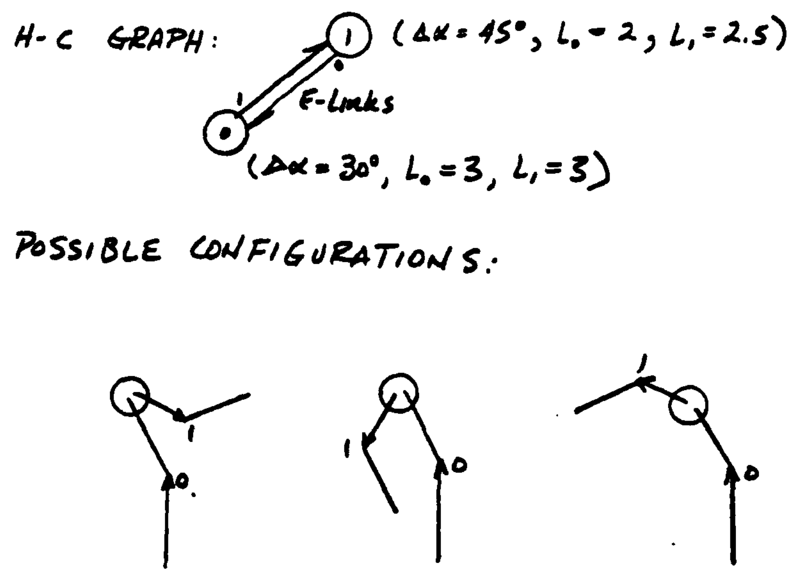
9) The primitive elements and relations used to encode objects should explicitly encode elements and relations required in the recognition process.

10) Models should be comparable without reparametrization.

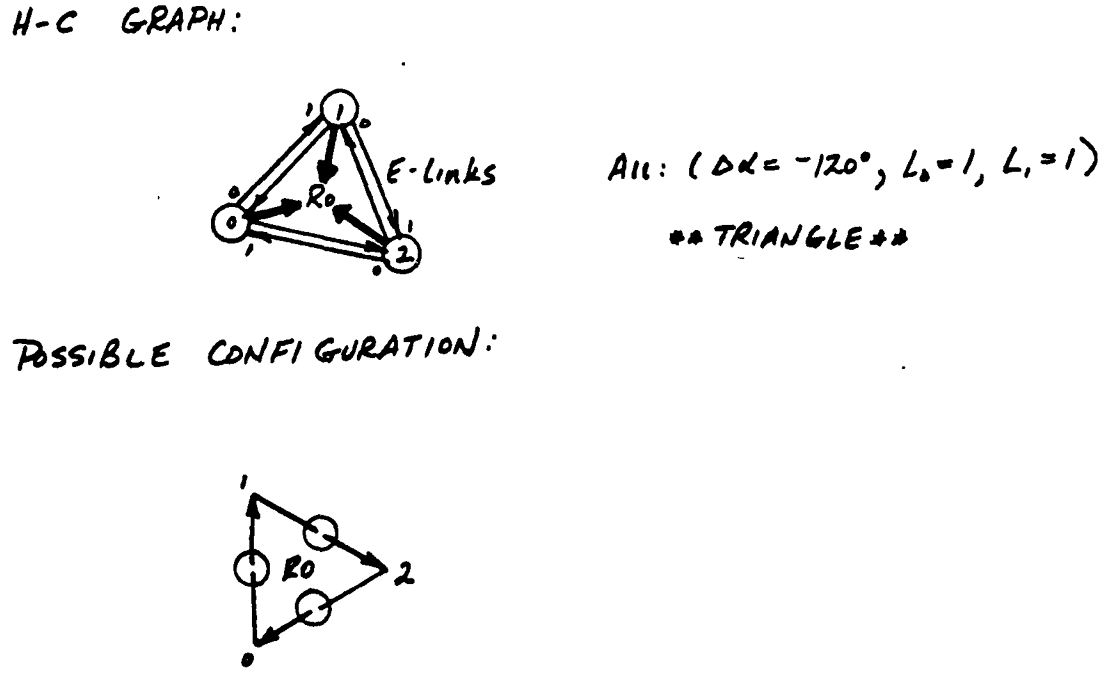
The semantic modeling approach satisfies (2) because complete semantic graphs need not be encoded, (5) and (9) because presumably ideal models would all contain critical relations and structure, and (10) because critical relations and structure will not change from one parametrization to another. The solid model approach satisfies (1) because complete dimensional information is stored, (2) because incomplete information is possible, (3) because all data can be stored in an averaged form (a "defocused", lower dimensional form), (5) only if an averaging process exists which can match models of differing size or dimension (i.e. (5) is not satisfied passively - a computational process is required), (9) if simple dimensional information is require only (i.e. (9) is not satisfied if we want complex topological relationships to be represented). Briefly, neither pure object centered solid representation nor semantic representation has all the features we desire for an object modeling scheme, although the solid modeling approach seems to be better. The generalized cylinder approach is better than either except that not all objects are amenable to it (i.e. for some objects like blocks major axis encoding does not instantiate important relationships).

We encode objects in a novel scheme based on the "half chunk" graphs produced by our depth-motion-labeling process (see Appendix D for a brief description of the labeling pass following depth-motion correlation). Recall that the basic half chunk (H-C) is a small section of object boundary curvature, which carries an array of properties describing the nature of the surface-surface interface locally. Dimensionally, a change in tangent angle and two lengths (lengths of the legs) are carried. In the original 2-d half chunk encoding, each H-C also carries a tangent and the coordinates at the center and end of both legs. These parameters are removed (tangent angle) or store in a separated structure (coordinates of the last known position for each H-C is stored in the XYZ atoms - these each index to a H-C atom, but are not used in our recognition algorithm - the XYZ atoms are kept as data for low level feature trackers).

For simple objects (Figure 5.1), the object H-C graph does not contain enough information to reconstruct a dimensional model of an object, however, when the object complexity increases just a bit (Figure 5.2 and 5.3), complete information in the form of constraint relations does exist. By constraint relations we mean that each H-C carries both dimensional constraints (delta angle and lengths), and relational constraints (X-link, E-link, and R-link bindings). On any reasonable 3-d object these constraints when taken in large enough subgroup sizes, completely describe the solid nature of an object. We call this form of representation "feature" centered as opposed to "object" center, the form commonly used for solid object representation. This form of modeling allows both 2-d (Figure 5.2), 3-d (Figure 5.3), and mixed dimensioned objects to be represented. Also, because any surface or boundary can be represented by a concatenation of half chunks, to any degree of precision, the requirement of exact representation can be achieved.

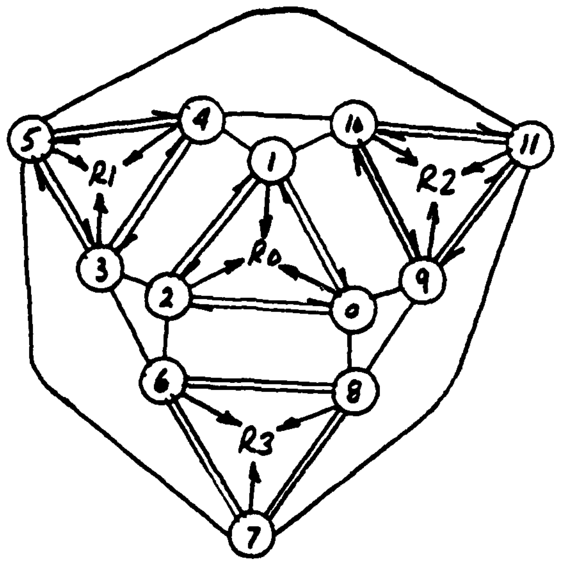


A Half Chunk Graph With Redundant Reconstructions
Figure 5.1



A 2-D Half Chunk Graph With A Single Reconstruction
Figure 5.2

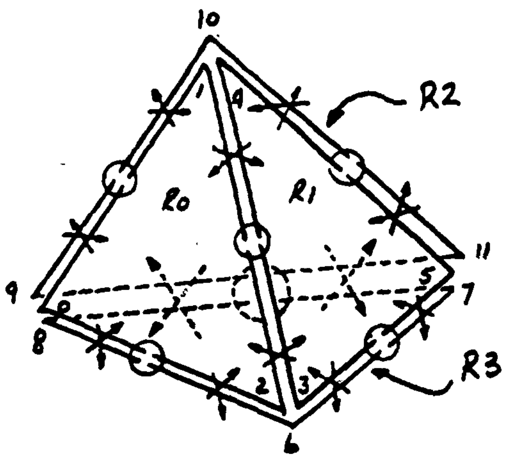
H-C GRAPH:



ALL: ($\Delta\alpha = -120°$, $L_0 = 2$, $L_1 = 2$)

** TETRAHEDRON **

POSSIBLE CONFIGURATION:



A 3-D Half Chunk Graph With A Single Reconstruction
Figure 5.3

Our method of encoding multiple objects uses a dictionary of universal half chunks which are referenced via tokens in all object H-C graphs. An object model consists of a sequence of disconnected 2-d and/or 3-d H-C graphs. An arbitrary number of individual objects are allowed in an object library. Library objects are compared to input objects (formed by the depth-motion-labeling correlator system) by producing and comparing series of subgraph histograms, beginning with histograms of basic H-C frequency. We find the histogram method allows partial object matching, matching of models composed at differing precisions, and rapid removal of unlikely objects from the matching set of objects. We believe that objects could actually by modeled by a series of subgraph histograms rather than by the original H-C graphs, thereby reducing recognition from a series traversal of graphs, to one of a series comparison of histograms. In this way, recognition of one object out of a library of many objects can be made to be significantly more computationally tractable.

5.2. A Common Basis for Object Graphs - Histograms

If we wish to compare graphs we can either compute some graph descriptor function (such as the number of atoms) for each of several graphs and compare the descriptor values, or we can traverse the graphs in some way, comparing graph elements serially, accumulating a match coefficient or cost (as done in depth and motion matching). If we take the traversal approach, we can either traverse the graphs in a breadth first or a depth first oriented way. Assume that we have a library of j object graphs each averaging m elemental atoms in size. If we wish to compare a new object encoded as a graph of n atoms to the library of objects, using the breadth or depth oriented search approach we require on the order of $j*m*n$ compare operations. To use the descriptor approach we need only $j*m+n$ operations or approximately $1/n$ times the traversal approaches. If the descriptors extracted from the object graphs are invariant we only require $j+n$ operations ($1/(m*n)$ times if j dominates, likely for large object libraries - $1/(j*m)$ times if n dominates, likely for small libraries of complex objects). Because we have been interested in methods which could be used with very large data bases we have been attracted to comparison by descriptor methods. Particularly we want descriptors that are either invariant or easily updated when new information appears (i.e. that do not require evaluation of all the information pertaining to a given object for each acquisition of new data).

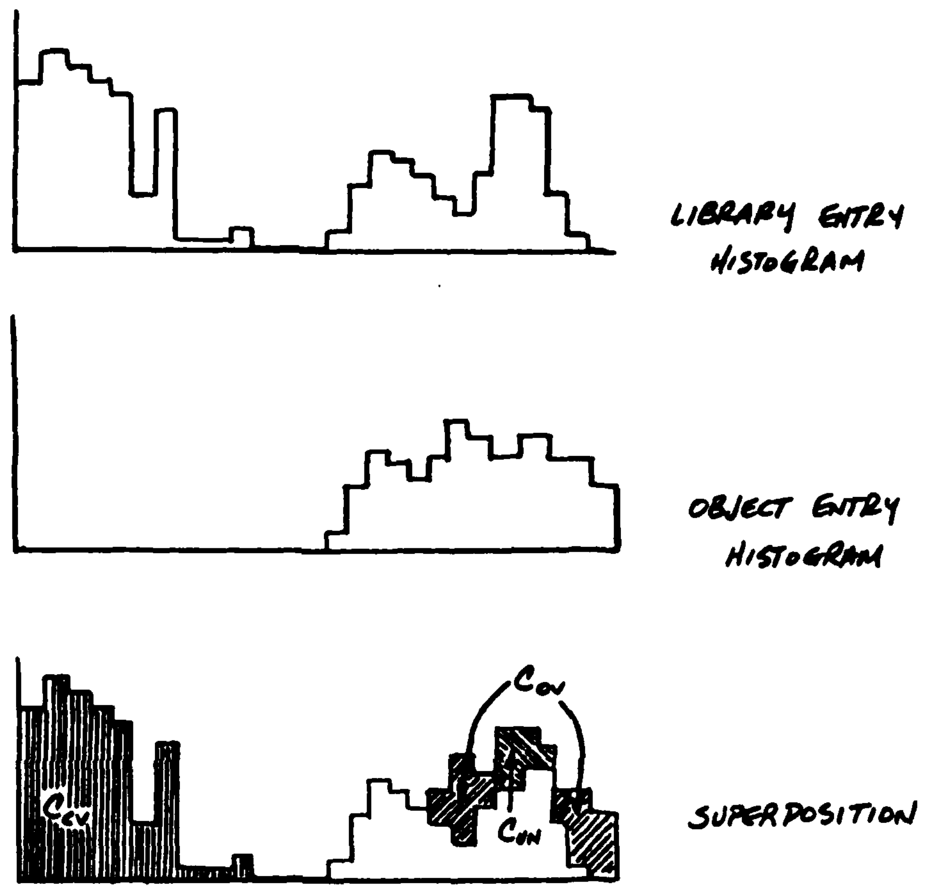
The type of graph descriptor chosen is based on critical subgraph histograms. Because one descriptor may not be powerful enough to separate all pairs of graphs, our method is based on comparison of families of descriptors, each accounting for higher order subgraphs. We first match each primitive half chunk in new object graphs to the half chunk component library. This library contains copies of all half chunks used in any graph currently encoded in the object data base. Any half chunks not having a match in the component library are entered. All half chunks are thus replaced by H-C tokens. The first order histogram of an object graph has one slot for each H-C in the components library. Each slot is incremented for an occurrence of the corresponding H-C token in the object graph.

To compute the second order histogram of an object graph we must generate all subgraphs with two atoms (or H-C's), centered at each atom. We make a sorted list of the subgraphs (to achieve an invariant ordering), collapse all identical entries into one, and use this as a component library. We may then compute a histogram using the uncollapsed list of binary subgraphs and the new second order components list. This operation may be performed repetitively for higher order subgraphs, however, each level is computationally more expensive. For arbitrary graphs computation becomes expensive quickly, however, because the H-C graph primitive has limits on link fan-out, it exibits more well behaved growth. We still do not find it practical to expand extensively beyond second or third order histograms. This corresponds to matching on the frequency of curvatures, pairs of curvatures, and trihedral vertices. Beyond the order three there is reason to believe that a traversal base technique might perform better computationally.

5.3. Histogram Comparison

Comparing object graphs has been reduced to comparing lists of histograms. To compare lists of histograms we need to address the problem of comparing single histograms, one from a new object graph and the other from a library of object histograms (or extracted from a library of object graphs). The simplest comparison method would be to accumulate a least-squares like difference cost between the two histograms (Figure 5.4 - $C_{ov} + C_{vm} + C_{cv}$). If this cost exceeds a limiting cost, the histograms are different, otherwise they are the same, perturbed by measurement noise. This method is undesirable because any new packet of views of an object is likely to be incomplete (unless we explicitly intended the packet as a learning set and therefore included sufficient information to build an adequate model).

We generally see only a few sides or discriminating features of an object during a recognition set, this being enough to separate a given object from our space of all objects. Except during initial object learning, we expect new object packets to be <u>subsets</u> of an object already encoded into our library. Therefore, matching error is only accumulated on histogram differences where new object features are not "covered" by old object library entries (Figure 5.4 - C_{MM}).



C_{OV} - OVERAGE COST

C_{UN} - UNDERAGE COST

C_{CV} - COVERED FEATURES COST

$C_{NM} = C_{OV} + C_{UN}$

Histogram Cost Definitions
Figure 5.4

We will evaluate all three histogram matching costs C_{ev}, C_{vw} , and C_{cv} even though the first alone is used to determine object matching. The other costs can be used heuristically in reducing search when libraries of objects get quite large. As long as C_{ev} is smaller than a preset limiting value, any mismatch is assumed to be due to information present in the new object histogram, not currently reflected in the library generated histograms. This new information is added to the library, refining the old object information, therefore allowing the system to "learn" about objects. On the other hand if the mismatch cost is greater than the limit, a new object entry must be formed.

This points to a very important phenomenon of "learning by similarity" systems. If the system is allowed to learn too fast (i.e. the mismatch cost limit is too high), all new data sets will be incorporated into one of a very few object models. If the system is constrained to learn very slowly (i.e. mismatching errors must be very small for equivalence), new data sets tend to form entirely new object models easily. We want practical systems to operate between these two extremes. Imagine each object packet generating a point in an object library space. For practical systems, we require a metric in this space which gives a low value between points associated with the same object, and significantly larger values between points associated with different objects. As a corollary, we expect better performance the larger the average distance between different object centers. Therefore, even if we have a good metric, we need to start the system with "good" (i.e. far separated) seed object examples.

5.4. Strategies for Constraining Search

In principle, we can form a series of histograms from any object graph, then compare this series with similarly generated series from other graphs to establish matching. This can be done between any input object graph and any graph in an object library. For large object data bases this is ridiculous because of computational constraints. We have programmed into our system several different ways for artificially constraining the object search (constraining the search without guaranteeing optimality). We may operate in new object mode, object verification mode, and constant computational effort mode. For all operating modes first order histograms for all objects in the library are precomputed and stored. Each new input object graph is processed individually and the library file updated accordingly.

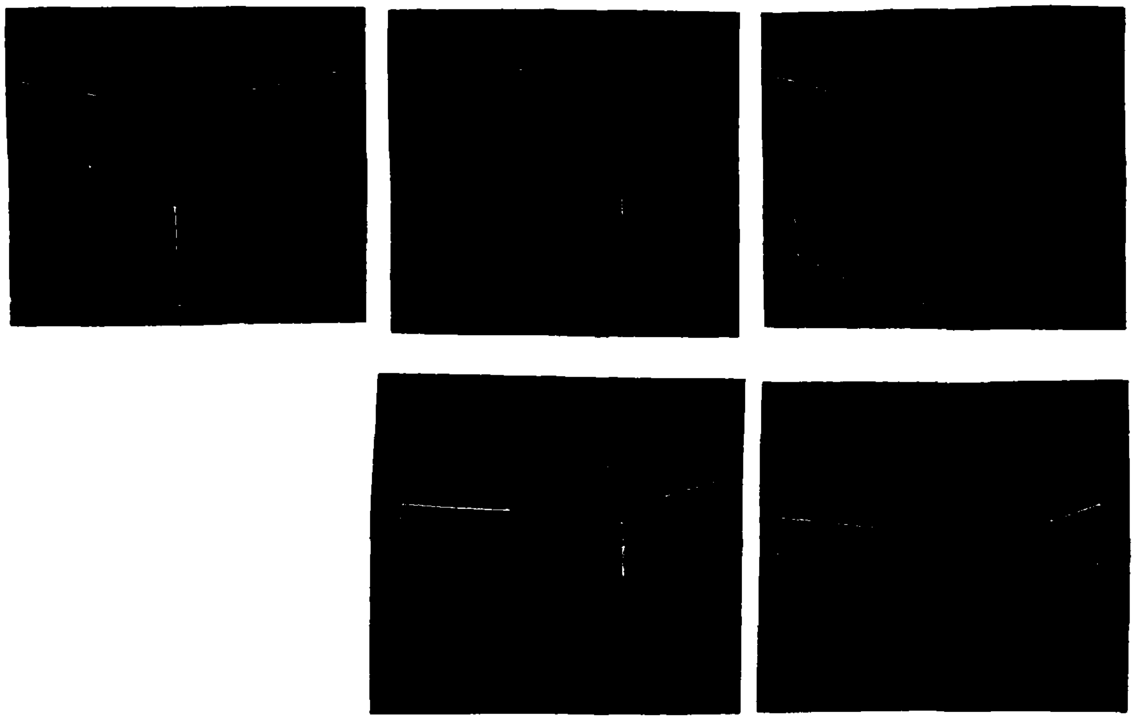
For new object mode, the first order histogram for the input graph is computed. This histogram is compared against all the precomputed first order histograms. If it is dissimilar from all the others, a new object entry (and associated histogram) is formed. Thus an new object is "learned". If only one histogram from the library is similar, a match is declared and the new object graph is concatenated to the graph data stored for the matched object (matched histogram is updated accordingly). If several histograms are similar, each is expanded to the second order, along with the one from the input object graph. The matching of the second order histograms proceeds as did the first. Expansion of higher order histograms continues until either a unique match is found or all potential matches are discarded (a new object entry is composed). This strategy allows new objects to be entered and old objects to be recognized. Computational load is roughly proportional to k*n where n is the number of objects in the library and k is related to average object complexity. Most of the work is done with the first order histograms, thus eliminating large portions of the search space after one level.

For object verification mode, we disallow any new object entries. Again the first order histogram for the object to be recognized is computed and compared to all first order histograms in the library. If we allow matching of no objects (i.e. this object is allowed to match none), the matching proceeds as in the new object mode case, except that no new objects are entered into the library when all matches fail. If we insist on a match, and we get to a level of comparisons (first order or higher) where no possible matches occur, we select the lowest cost match (using C_{ov} as the index of cost) as a correct match. This basically forces the selection of an object. This matching mode will have better performance in error prone environments, at the expense of disallowing new object model building. Computationally, this method of operation is equivalent to the new object mode.

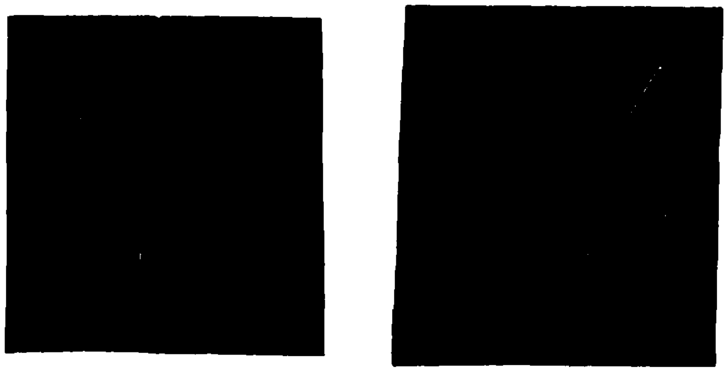
In the constant computational effort mode matching, we constraint object space searching artificially by examining matching costs (using $\mathcal{C}_{OV}$ and $\mathcal{C}_{UN}$). The heuristic used does not guarantee optimality, but generally yields good results for large object libraries. We expand the first order histogram for the input object graph, and compare to the precomputed histograms. We eliminate all histograms not considered similar. We treat the single and null match as previously. However, in the case of the multiple match we disallow all but the m best matches (m is a specified constant, and best to worst is ranked using the $\mathcal{C}_{MM}$ cost - similarity is still based on thresholding the $\mathcal{C}_{OV}$ cost). If after selection, multiple matches still abound, we descend to second order histogram based comparisons. If these yield multiple matches, we examine the third order matches. The parameter m is roughly equivalent to the depth of our most intensive search in histogram orders. In this way we can artificially limit computation during object matching. Intuitively, it seems reasonable that recognition can be accomplished for most machine parts with limiting search to three histogram orders - curvatures, paired curvatures, and trihedral vertices.

Figures 5.5 and 5.6 show the objects recognized and entered in our test object libraries generated by runs on the data set shown in figures 4.14-4.19. Note that while some data will be missing for specific instances of an object, the data that is present has an invariant relationship to model data (with some measurement error).

Squares

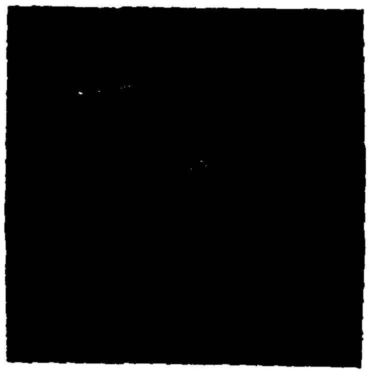


Curved Wedges

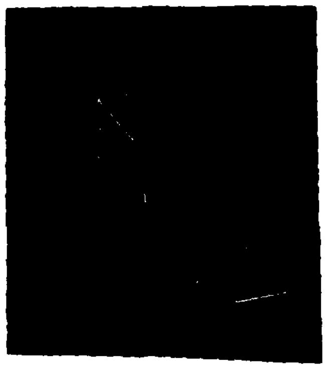


Object Fragments - I
Figure 5.5

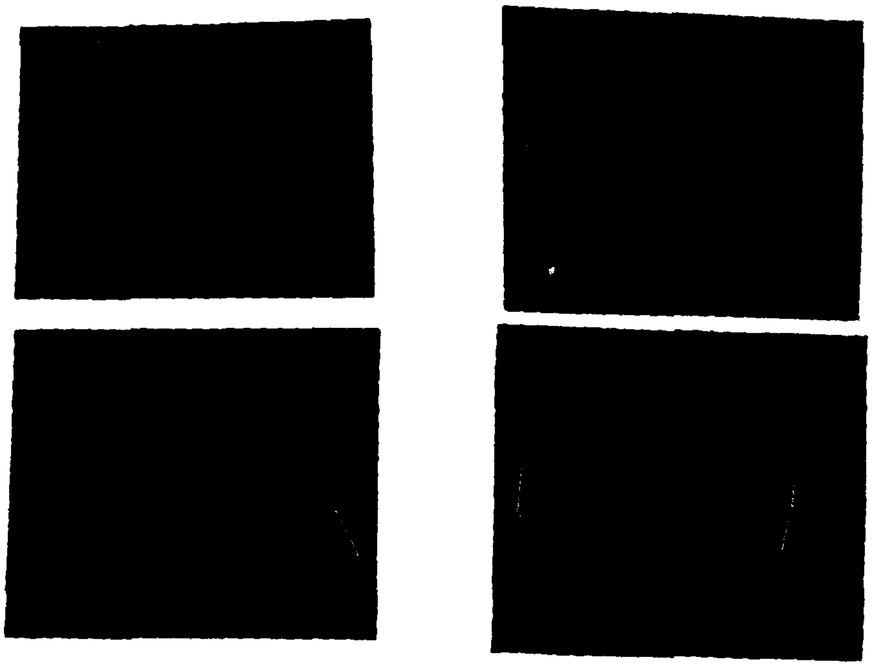
Pulley



Stapler



Piston Rod



Curved Cutout

Object Fragments - II
Figure 5.6

5.5. Using Histograms as the Models

In the current implementation of the object modeling system, we have a basic half chunk component library ordered by intrinsic half chunk value (sorted on all H-C properties, inducing an invariant order). Each H-C is given a token value at the time of entry into the component library. All H-C graphs for objects are re-encoded using this token value in lieu of actual half chunks. Prior to object recognition, the first order histograms for each object model are computed from these H-C graphs, stored in the object library. As required, higher order histograms are computed during object matching sequences. Because we have found that generally only limited levels of histogram orders really need to be traversed for most sets of objects, it seems quite reasonable to eliminate actual graph storage in the future.

Modeling objects as histograms, or as parameters which characterize (such as μ and σ) histograms is quite appealing, because it allows us to unify shape and texture recognition techniques. We have pointed out elsewhere (Chapter 3) that texture is best modeled as an overdose of shape information. Using histograms of edge and edge connector densities and directions Marr has described many elemental textures. Using our histogram techniques we can describe object forms. While histogram segmentation across whole images many be questionable, most elemental difference-based edge detection schemes can be posed as local histogram characterization and matching.

The shape histograms herein described may easily be incrementally formed (i.e. updated to reflect new information), if we fully expand the components library to all possible H-C primitives. This set of low level primitives then becomes our basic object alphabet. Individual objects become statistical "words" composed of these letters without order (first order histograms), with binary ordering constraints (second order), etc. We really never need to encode graphs to encode object structure.

5.6. Conclusion

We have described a modeling system which is "feature" centered rather than "object" centered. This method has the good features of semantic net models and dimensional models combined. We have discussed several model matching methods which allow for automatic model building and model verification.

The method we propose is based on objects modeled as 2-d and 3-d half chunk graphs. These graphs are relational networks of elemental curvature samples. The basic model is free of scale, coordinate system, or rotation dependencies (however, XYZ atom information may be incorporated to bind the half chunks to a specific three dimensional space). This allows model comparisons without any reparametrization.

To speed the recognition process, allow for matches on partial data, and organize search, we have describe a histogram-based object graph matching scheme. This method may be extrapolated to actual object modeling by histograms.

Cnapter 6

Contributions And Future Work

6.1. Contributions

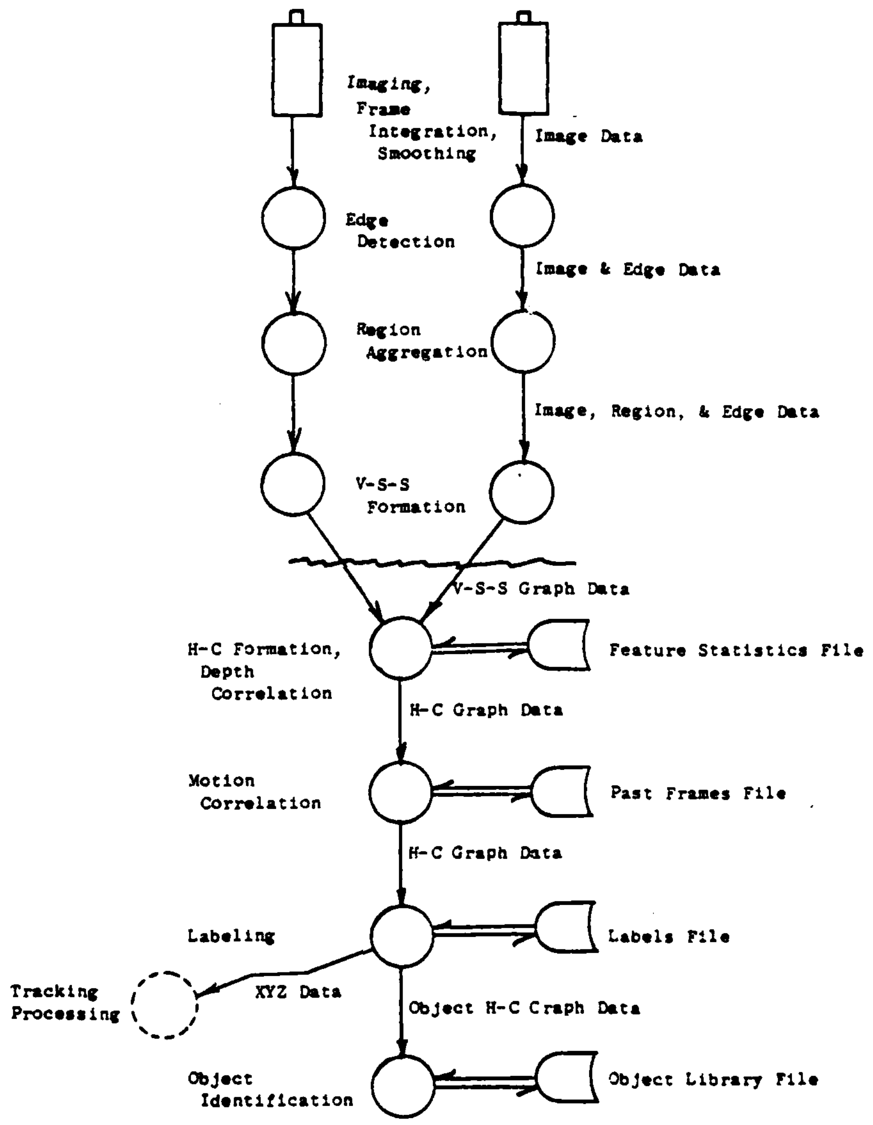
This work first and foremost is an attempt at constructing a complete visual understanding system: from initial measurement and segmentation to object learning and recognition.

We have contributed two new edge detection schemes (Chapter 2), a new region-based edge reinforcement process (Chapter 3), and a new algorithm for boundary and vertex codification (Chapter 3).

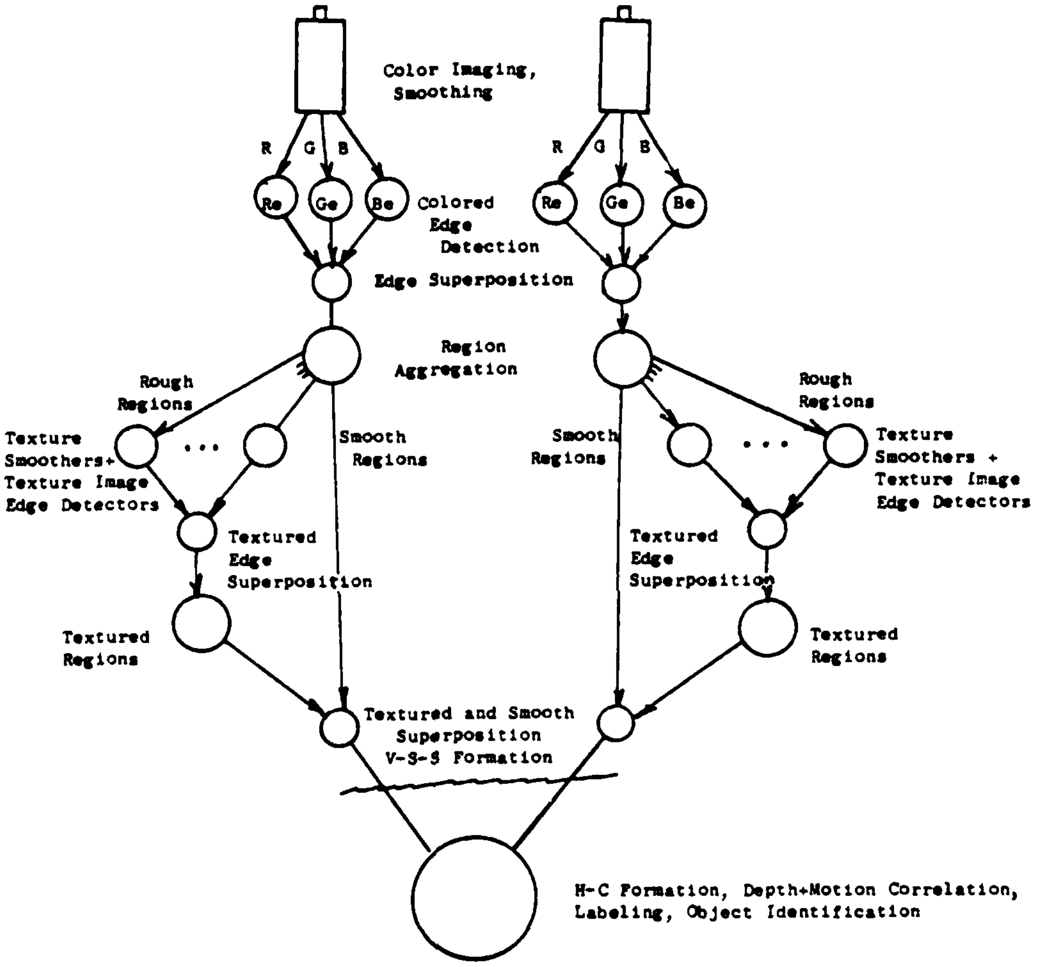
The most significant contribution is the half chunk, half chunk graph, and the histogram-based graph comparison method (for object library maintainance). The H-C graph is significant because it points the towards "feature" centered object models instead of "object" centered models (Chapter 4). The histogram-based modeling system is significant (Chapter 5) because it point towards a method of processing visual data that could unify object recognition, texture discrimination, edge detection, and region aggregation (all as histogram characterization processes).

A smoothing method (Appendix B), a method for threshold detection (Appendix C), and a feature based labeling scheme (Appendix D), have also been developed.

Figures 6.1 and 6.2 show a block diagram of our system operating in non-textured black and white environments and in colored-textured environments.



The Basic Vision System
Figure 6.1


Color Imaging, Smoothing
R
G
B
Re
Ge
Be
Colored Edge Detection
Edge Superposition
Region Aggregation
Rough Regions
Texture Smoothers+ Texture Image Edge Detectors
Smooth Regions
Textured Edge Superposition
Textured Regions
Textured and Smooth Superposition V-S-S Formation
H-C Formation, Depth+Motion Correlation, Labeling, Object Identification

System Enhanced to Operate in
Colored-Textured
Environments
Figure 6.2

6.2 Future Work

The greatest speed and performance improvements will come from work on the first few processing steps, i.e. edge measurements and region-based processing. These both now work with 60000 pixels per image and this resolution will grow. We now require approximately 20 minutes per frame for these steps and only 5 minutes per frame for all our graph-based manipulations (this is even more significant if you consider that the 20 minute phases are hand coded to execute as fast as possible, whereas the 5 minute phases are implemented quite inefficiently). By the time we have reduce our processing to edges, we have nominally 2000 points. After V-S-S formation hundreds of things remain. By the time of object identification we very rarely have more than 10 individual structures left to process.

Reliability is still limited most by how well initial significant differences can be measured. Cameras need better dynamic range, spatial resolution, and distortion characteristics (spatial and intensity ranges).

We see no good reason why edge information cannot be directly encoded into half chunk form without the intermediate stage of the V-S-S graph. We also see no reason why more relaxation-based processing could not be incorporated into a single program, allowing richer interaction between the high and low levels of the system (particularly, with respect to depth, evidence indicates that humans can use depth information at a lower level for segmentation aids prior to object formation).

We feel that distributed computing architectures would greatly improve the real time performance especially at the low end where help is critical. There should never be any reason for a general purpose serial machine to evaluate a function over every pixel site.

We feel this effort towards a bottom-up based system is a mixed success. We have extensively tested our methods for edge and region detection and formation. These are quite successful. Our texture processing has been done for only a few examples and runs too slowly for extensive testing. It currently knows about edge densities with and without directions. The graph matching methods for depth and motion works well for objects having some sharp curvature discontinuities but not so well for things with very regular structure, like spheres. For these, a "half chunk"-like structure for expressing centers of object symmetry are a possible solution. We would propose a system which uses both boundary generated half chunks and symmetry axis generated half chunks. Also things with no regular dimensional structure, like trees, cannot be modeled. This problem would probably best be solved by incorporating a half chunk based system within a more global semantic net based system (possibly coupled to a natural language system). It has been difficult to process enough data for more than about eight objects, so the modeling system has not been extensively tested. We, therefore, can only speculate on performance with very large object data bases.

Bibliography

[1] Aggarwal, J. and R. Duda, Computer Analysis of Moving Images, Tech. Rep. 161, Univ. of Texas (Oct 1974).

[2] Baker, H., "Three-Dimensional Modeling", Pr. 5th. IJCAI (1977), pp. 649-655.

[3] Barrow, H. and J. Tenenbaum, Recovering Intrinsic Scene Characteristics from Images, Stanford Res. Inst. Tech. Note 157 (April 1978).

[4] Baumgart, B., Geometric Modeling for Computer Vision, Stanford AI Memo AIM-249 (1974).

[5] Bennett, J. and J. Mac Donald, "On the Measure of Curvature in a Quantized Environment", IEEE Trans. on Comp. C-24, 8 (1975), pp. 803-820.

[6] Brice C. and C. Fennema, "Scene Analysis Using Regions", Art. Intel. 1 (1970), pp. 205-226.

[7] Burr, D., A System for Stereo Computer Vision with Geometric Models, Ph.D. Thesis, CSL, Univ. of Ill. (1977).

[8] Chang, Y. Machine Perception of Objects with Curved Surfaces, Ph.D. Thesis, CSL R-641, Univ. of Ill. (1974).

[9] Deutsch, E. and J. Fram, "A Quantitative Study of the Orientation Bias of Some Edge Detector Schemes", IEEE Trans. on Comp. C-27, 3 (March 1978), pp. 205-213.

[10] Duda, R. and P. Hart, Pattern Classification and Scene Analysis, Wiley (1973).

[11] Dudani, S. and C. Clark, "Vertex-Based Model Matching", Pr. Symposium on Current Math. Problems in Image Sci., Monterey, Cal. (Nov 1976).

[12] Dunn, J., "Group Averaged Linear Transforms That Detect Corners and Edges", IEEE Trans. on Comp. C-24, 12 (1975), pp. 1191-1201.

[13] Falk, G. "Scene Analysis Based on Imperfect Edge Data", Pr. 2nd. IJCAI (1971), pp. 8-16.

[14] Feldman, J. and Y. Yakimovsky, "Decision Theory and Artificial Intelligence: I. A Semantics-Based Region Analyzer", Art. Intel. 5 (1974), pp. 349-371.

[15] Feng, F. and T. Pavlidis, "The Generation of Polygonal Outlines of Objects from Gray Level Pictures", IEEE Trans. on Circuits and Systems CAS-22, 5 (May 1975), pp. 427-439.

[16] Feng, H. and T. Pavlidis, "Finding 'Vertices' in a Picture", Comp. Graph. and Image Proc. 2 (1973), pp. 103-117.

[17] Freuder, E., Synthesizing Constraint Expressions, MIT AI Memo 370 (July 1976).

[18] Gennery, D., "A Stereo Vision System for an Autonomous Vehicle", Pr. 5th. IJCAI (1977), pp. 576-582.

[19] Griffith, A., "Mathematical Models for Automatic Line Detection", J. of ACM 20, 1 (Jan 1973), pp. 62-80.

[20] Hanson, A., E. Riseman and P. Nagin, "Region Growing in Textured Outdoor Scenes", Pr. 3rd. Milwaukee Symposium on Automated Comp. and Control (1975), pp. 407-417.

[21] Horn, B., "Understanding Image Intensities", Art. Intel. 8 (1977), pp. 201-231.

[22] Hueckel, M., "An Operator Which Locates Edges in Digitized Pictures", J. ACM 18, 1 (Jan 1971), pp. 113-125.

[23] Hueckel, M., "A Local Operator Which Recognizes Edges and Lines", J. ACM 20, 4 (Oct 1973), pp. 634-647.

[24] Hueckel, M., Erratum for [23], J. ACM 21, 2 (April 1974), p. 350.

[25] Huffman, D., "Impossible Objects as Nonsense Sentences", Mach. Intel. 6, Edinburgh Univ. Press (1971).

[26] Huffman, D., "Realizable Configurations of Lines in Pictures of Polyhedra", Mach. Intel. 8, Edinburgh Univ. Press (1977).

[27] Jain, R., D. Militzer and H. Nagel, "Separating Non-Stationary Scene Components in a Sequence of Real World TV Images", Pr. 5th. IJCAI (1977), p. 612.

[28] Jacobus, C. and R. Chien, *Variable Neighborhood Computations in Scene Analysis*, CSL T-60, Univ. of Ill. (1973).

[29] Krakauer, L., *Computer Analysis of Visual Properties of Curved Objects*, Ph.D Thesis, MIT MAC TR-32 (1971).

[30] Lindgren, B. *Statistical Theory*, Macmillan (1976).

[31] Marr, D., *On the Purpose of Low-Level Vision*, MIT AI Memo 324 (Dec 1974).

[32] Marr, D., *The Low-Level Symbolic Representation of Intensity Changes in an Image*, MIT AI Memo 325 (Dec 1974).

[33] Marr, D., *The Recognition of Sharp, Closely Spaced Edges*, MIT AI Memo 326 (Dec 1974).

[34] Marr, D. and T. Poggio, *From Understanding Computation to Understanding Neural Circuitry*, MIT AI Memo 357 (May 1976).

[35] Marr, D., *Early Processing of Visual Information*, MIT AI Memo 340 (1975).

[36] Marr, D., *Cooperative Computation of Stereo Disparity*, MIT AI Memo 364 (June 1976).

[37] Marr, D. and H. Nishihara, *Representation and Recognition of the Spatial Organization of Three Dimensional Shapes*, MIT AI Memo 377 (Aug 1976).

[38] Marr, D., *Analysis of Occluding Contour*, MIT AI Memo 372 (Oct 1976).

[39] Marr, D., G. Palm and T. Poggio, *Analysis of a Cooperative Stereo Algorithm*, MIT AI Memo 446 (Oct 1977).

[40] Martelli, A., "An Application of Heuristic Search Methods to Edge and Contour Detection", Comm. of the ACM 19, 2 (Feb 1976), pp. 73-83.

[41] McKee, J. and R. Aggarwal, "Finding the Edges of the Surfaces of Three-Dimensional Curved Objects by Computer", Pattern Recogn. 7 (1975), pp. 22-52.

[42] McKee, J. and R. Aggarwal, "Computer Recognition of Partial Views of Curved Objects", IEEE Trans. on Comp. C-26, 8 (Aug 1977), pp. 790-800.

[43] Mero, L. and Z. Vassy, "A Simplified and Fast Version of the Hueckel Operator for Finding Optimal Edges in Pictures", Pr. 4th IJCAI (1975), pp. 650-655.

[44] Nevatia, R. and T. Binford, "Description and Recognition of Curved Objects", Art. Intel. 8 (1977), pp. 77-98.

[45] Ohlander, R., Analysis of Natural Scenes, Ph.D. Thesis, CMU (April 1975).

[46] O'Gorman, F. and M. Clowes, "Finding Picture Edges Through Collinearity of Feature Points", Pr. 3rd. IJCAI (1973), pp. 543-555.

[47] Perkins, W. and T. Binford, "A Corner Finder for Visual Feedback", Comp. Graph. and Image Proc. 2 (1973), pp. 355-376.

[48] Perkins, W., A Model-Based Vision System for Industrial Parts, GMR-2410 (June 1977).

[49] Potter, J., "Scene Segmentation Using Motion Information", Comp. Graph. and Image Proc. 6 (1977), pp. 558-581.

[50] Ramer, E., "The Transformation of Photographic Images into Stroke Arrays", IEEE Trans. on Circuits and Systems CAS-22, 4 (April 1975), pp. 363-374.

[51] Roberts, L., "Machine Perception of Three-Dimensional Solids", Optical and ElectroOptical Information Processing, Tippett ed., MIT Press (1965), pp. 159-197.

[52] Rosenfeld, A. and M. Thurston, "Edge and Curve Detection for Visual Scene Analysis", IEEE Trans. on Comp. C-20, 5 (May 1971), pp. 562-569.

[53] Rosenfeld, A., M. Thurston and Y. Lee, "Edge and Curve Detection: Further Experiments", IEEE Trans. on Comp. C-21, 7 (July 1972), pp. 677-714.

[54] Rosenfeld, A., A. Hummel and S. Zucker, "Scene Labeling by Relaxation Operations", IEEE Trans. on Systems, Man, and Cyber. SMC-6, 6 (June 1976), pp. 420-433.

[55] Selander, J. M., "Cost Based Graph Matching", Personal communications, Sept 1979.

[56] Schacter, B., L. Davis and A. Rosenfeld, "Scene Segmentation by Cluster Detection", SIGART Newsletter 58, ACM (June 1976), pp. 16-17.

[57] Shapira, R. and H. Freeman, "Reconstruction of Curved-Surface Bodies from a Set of Imperfect Projections", pr. 5th. IJCAI (1977), pp. 628-634.

[58] Shirai, Y., "Edge Finding, Segmentation of Edges and Recognition of Complex Objects", Pr. 4th. IJCAI (1975), pp. 674-681.

[59] Shneier, M., "Recognition Using Semantics Constraints", Pr. 5th. IJCAI (1977), pp. 585-589.

[60] Tenenbaum, J. and H. Barrow, "Experiments in Interpretation-Guided Segmentation", Art. Intel. 8 (1977), pp. 241-274.

[61] Tomita, F., M. Yachida and S. Tsuji, "Detection of Homogeneous Regions by Structural Analysis", Pr. 3rd. IJCAI (1973), pp. 564-571.

[62] Underwood, S. and C. Coates, "Visual Learning from Multiple Views", IEEE Trans. on Comp. C-24, 6 (June 1975), pp. 651-661.

[63] Waltz, D., Generating Semantic Descriptions from Drawings of Scenes with Shadows, Ph.D. Thesis, MIT AI TR-271 (1972).

[64] Waltz, D., "A Parallel Model for Low-Level Vision", (1977).

[65] Winston, P., Learning Structural Descriptions from Examples, Ph.D Thesis, MIT AI TR-231 (1970).

[66] Yakimovsky, Y., "Boundary and Object Detection in Real World Images", J. of ACM 23, 4 (Oct 1976), pp. 599-618.

[67] Zahn, C., "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters", IEEE Trans. on Comp. C-20, 1 (Jan 1971), pp. 68-86.

Appendix A

Edge Detectors

A.1. Hueckel Edge Detector

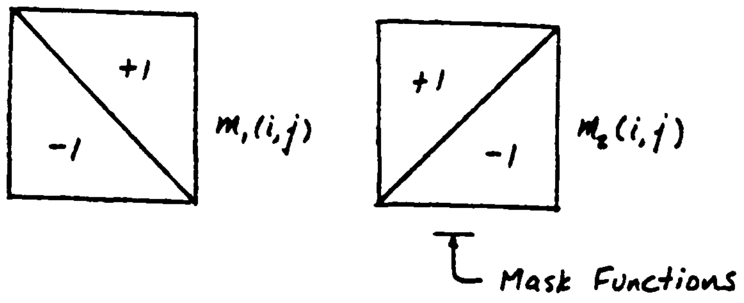
This edge detector is a direct implementation in BLISS-10 of the algorithm in [23] Appendix A, with fixes indicated in [24]. The operator is designed to work on a digitized disk of radius 4.5 (diameter of 9 cells). It allows detection of steps, dark stripes, and/or light stripes. In these experiments, we have restricted the algorithm to steps.

The operator attempts to explain all activity in a disk in terms of the superposition of a set of orthogonal functions. The intensity function is expressed as a series expansion in terms of the orthonormal set. A least squares criterion between the actual intensity function and the orthonormal expansion is used to compute, approximately, the coefficients for each term. From this the edge type, position, and direction is computed. For complete details the reader is referred to the literature.

A.2. Psuedo-Hueckel Detector

This operator is based in the same sort of mathematics as is the original Hueckel operator. The intensity variation in a square neighborhood of 9x9 is expressed as a superposition of orthonormal functions. In [43], it is shown that for detecting the direction (α) of any step in a window the functions of Figure A.1 are sufficient.

A.2.1. $$\frac{\sum_i \sum_j g(i,j)\, m_2(i,j)}{\sum_i \sum_j \underbrace{g(i,j)}_{\text{Image Function}}\, m_1(i,j)} = \tan\left(\alpha - \frac{3}{4}\pi\right)$$



Orthonormal Set For Direction Computation
Figure A.1

Our simple refinement to the procedure in [43], to allow detection in non-binary pictures, consists of computing the minimum and the maximum in the window, while computing the components of the gradient. We then select a threshold of T=(MAX-MIN)/2+MIN. [43] suggests that with such a threshold T, we can compute the edge line (i.e. the edge position), given the direction (α) and the number of cells greater than, or less than the threshold. While this is true, we found that it was actually easier to implement the position computation by fitting the least squares line,

A.2.2. $$y = \frac{\overline{xy} - \bar{x}\,\bar{y}}{\overline{x^2} \quad \bar{x}^2}\, x + \frac{\overline{x^2}\,\bar{y} - \bar{x}\,\overline{xy}}{\overline{x^2} - \bar{x}^2}$$

or

A.2.3. $$x = \frac{\overline{xy} - \bar{x}\,\bar{y}}{\overline{y^2} - \bar{y}^2}\, y + \frac{\overline{y^2}\,\bar{x} - \bar{y}\,\overline{xy}}{\overline{y^2} - \bar{y}^2}$$

using the points where the intensity flips from one side of the threshold to the other side of the threshold.

This modified operator is significantly faster than Hueckel's and seems equally sensitive. Each pixel in the square area is accessed two times. Once to compute MIN, MAX, and gradient components. Then a second time to acquire threshold crossover points, therefore complete the edge line information.

A.3. Linear Difference Detector

This operator was used by [7] in his edge detection apparatus, and was for us a jumping off point in our edge detection experiments. Our difference-based system is really an array of linear difference detectors with a better peak selection algorithm.

A six cell strip (vertical or horizontal) is used to compute a difference (three positive, three negative). This difference is compared to its nearest neighbors (Burr used the neighbors on both sides, we use several neighbors on each side). If it is a relative minimum or maximum, an edge is marked in its position. Burr went one step further. He examined the difference in the orthogonal direction to compute an angle for the edge (as in the psuedo-Hueckel operator). We simply mark vertical or horizontal edges.

A.4. Rosenfeld Non-Linear Edge Detector

This is a direct implementation of the edge detection method described in [52] and [53]. This system computes an array of averaged pictures, four in all for our implementation. The block sizes used in the averaging operations are 2x2, 4x4, 8x8, and 16x16. Rosenfeld also suggested 1x1 and 32x32, however we found these to be unnecessary (1x1 is smaller than the quickest step transition possible from our vidicon system, 32x32 is larger than required to average any noise pattern in our tests).

Then differences are formed in the vertical and horizontal directions (Rosenfeld suggests diagonals, but for computational efficiency sake, we use only two directions). At each averaged size, we used maxima suppression to remove (set to zero) all points that are not maxima or minima within one averaged block area. Additionally, the differences not suppressed must be greater than a confidence T.

We now examine the differences with respect to those in other averaged sizes. We select the difference for which then next smaller size does not give a significantly higher absolute difference. If E(i) is a difference for the averaged size i x i and E(i/2) is the maximum difference for the next level smaller (the maximum non-suppressed difference in the area covered by the i x i sized neighborhood at the (i/2) x (i/2) sized level), then "significantly higher" means:

```
A.4.1.  E(i) < aE(i/2) < (a**2)E(i/(2**2)) <
                 ... < (a**n)E(i/(2**n))

        but, E(i/(2**n)) > aE(i/(2**(n+1)))

        where, a_3/4
```

An edge is marked in the position of this selected difference. The (2**n) term will never exceed 8 (n=3), in our implementation. At n=3, the selection process automatically selects the difference (i.e. (i/(2**(n+1))) is never evaluated when n=3).

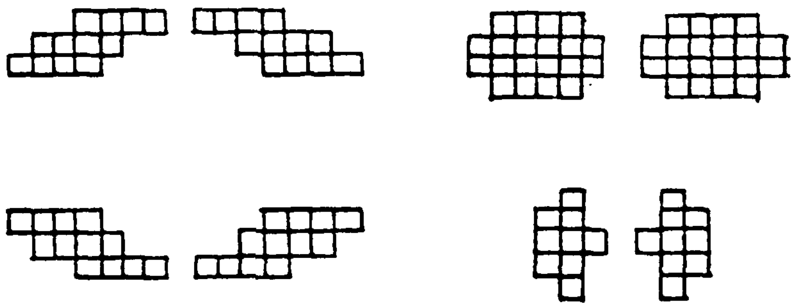
For further details we refer the reader to [52] and [53].

A.5. Yakimovsky Edge Detector

We have adopted the edge strength function proposed by [66], but not the region growing algorithm. This is done because we wish to examine edge detection techniques, not region-based processing. (Although, the region-based techniques may be interesting in comparison to those described in Chapter 3. There are similarities between our method of expanding edge neighborhoods for string and vertex detection and the growing rules used by Yakimovsky. By not being constrained to be one pass, our system is capable of doing better in some cases.)

We have simply coupled Yakimovsky's edge strength function to a non-maxima suppressing edge detection scheme. The strength function is given in Chapter 2, for reference to the more standard expressions for variance and mean comparisons.

The suppression algorithm used is like the one previously described for the linear difference detector. We scan the operator neighborhoods (four suggested in [66] - see figure A.2) horizontally and vertically, marking edges at each point where the output is maximum within one operator neighborhood area. The only complication is that we select our result from the operator having the largest strength, and suppress the others within the same area. Yakimovsky seems to be suggesting that several differently shaped neighborhoods might allow object shapes to be followed more affectively, considering that his technique requires a relatively large number of samples (to compute the variance accurately enough). In fact, from a purely statistical point of view it can be argued that even larger neighborhood sizes might be desirable.



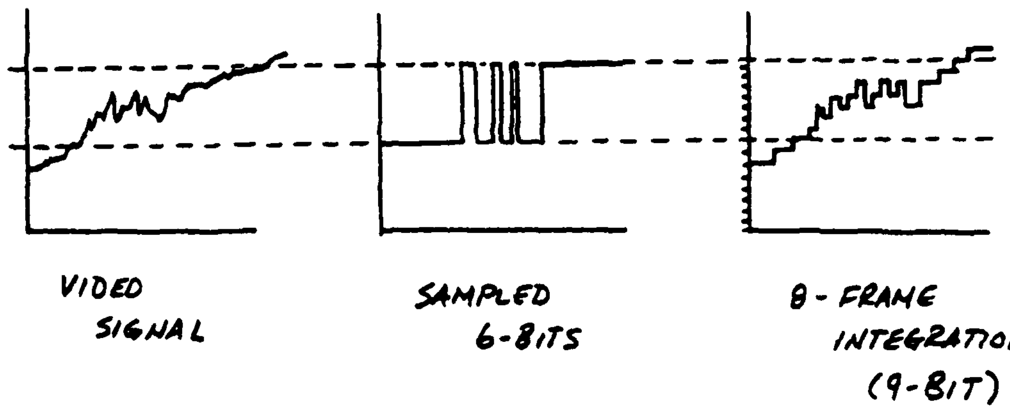
Yakimovsky's Edge Detection Neighborhoods
For Horizontal Edges
Figure A.2

Appendix B

Image Smoothing and Enhancement

B.1. Image Averaging

The basic images taken from the C.S.L. silicon vidicon imaging system are 252x238 six bit pixels each, plus or minus one level. For low contrast images, only 64 levels is not adequate (especially when performing gradient related operations). We have remedied this problem by averaging eight 252x238x6 bit images to form a 252x238x9 bit one. When the vidicon signal is near a quantization level, small noise variations can cause plus or minus one level digitization errors (the analog signal in our system can be modeled by the sum of the "ideal" signal + N(0.0,~0.5) - see Figure B.1). If we sum several images together, we can effectively get new levels "between" the old ones in single images, thus a finer quantization of the vidicon signal. This only works because there is additive noise in the vidicon image signal.



Vidicon Signals, Quantizations, and Averages
Figure 8.1

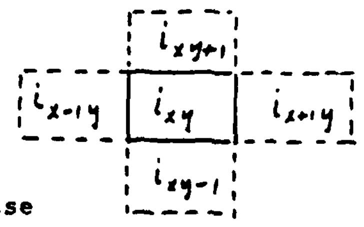
8.2. Image Smoothing

We can further improve image quality by smoothing. This generally consists of replacing every pixel by the average of all the pixels in a given neighborhood. While this does improve "noise" variations (by replacing a random variable with μ and σ by a new random variable with μ and $\frac{\sigma}{\sqrt{n}}$), it also blurs genuine signal variations.

To get around this problem, we designed a non-linear smoother based on the maximum likelihood decision between:

$$H_0: i_{xy} - i_{x\pm1\, y\pm1} \in N(0,\sigma)$$
$$H_1: i_{xy} - i_{x\pm1\, y\pm1} \notin N(0,\underset{\uparrow}{\sigma})$$

Estimated Noise Distribution



$\therefore$

i_{xy} is set to:

$$\frac{w(+1,0)\,i_{x+1y} + w(-1,0)\,i_{x-1y} + w(0,+1)\,i_{xy+1} + w(0,-1)\,i_{xy-1} + i_{xy}}{w(+1,0) + w(-1,0) + w(0,+1) + w(0,-1) + 1}$$

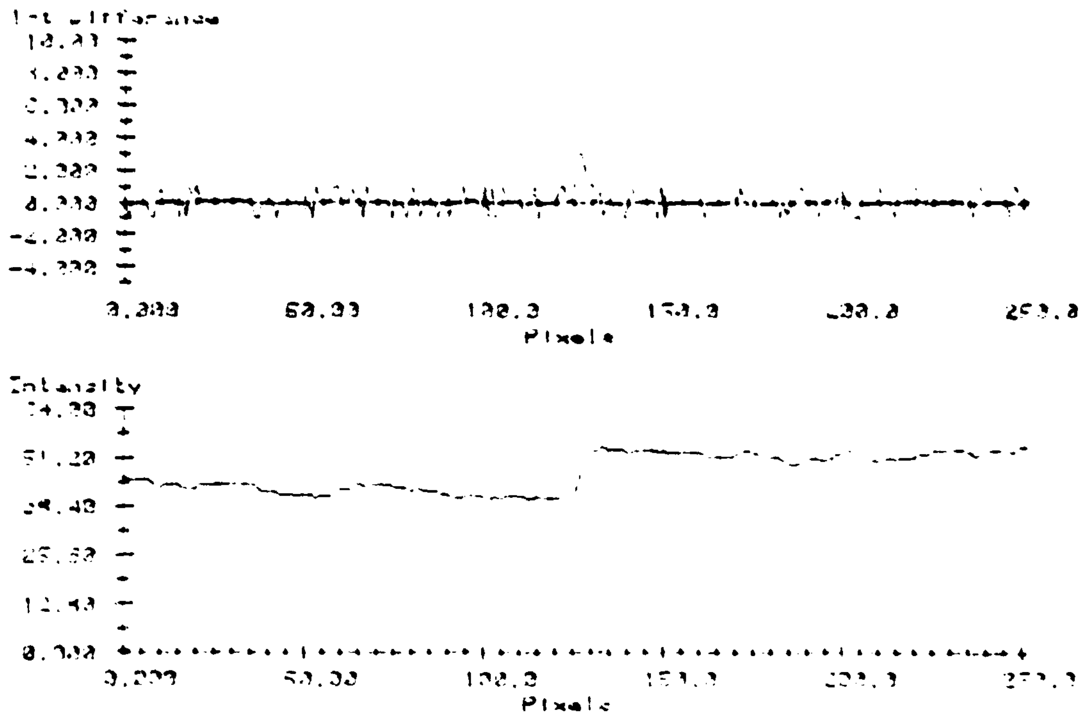
Where:

$$w(k,j) = \int_{i_{x+k\, y+j} - i_{xy}}^{i_{xy} - i_{x+k\, y+j}} \frac{1}{\sqrt{2\pi}\,\sigma}\, e^{\left[\frac{-1}{2\sigma^2} x^2\right]} dx$$

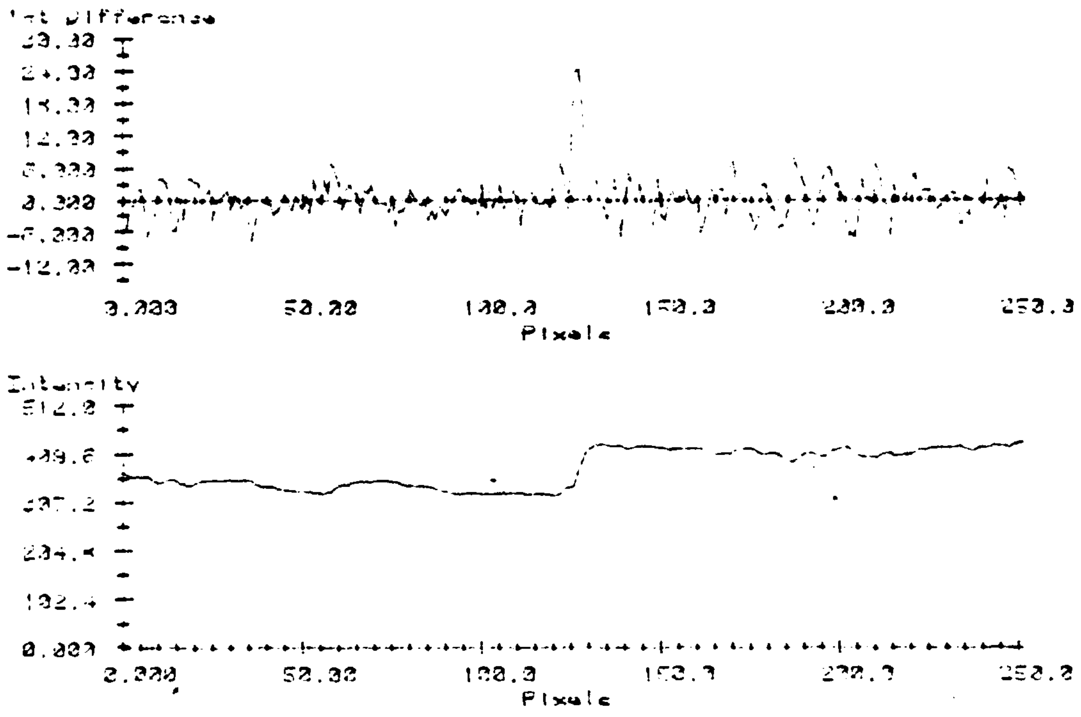
This decision is evaluated about each pixel for each of its adjacent neighbors, iteratively for several whole picture cycles. We have experimented with several methods for cutting off the iteration process. The first, and simplest, is to iterate a fixed number of times. For our typical images we have found 5 iterations to be adequate. Second, we have allowed iterations to continue until the overall σ of the image fails to become smaller, indicating signal degradation. This method, for typical images, runs between 3 and 6 cycles before stopping. The last method used stops cycling when σ, for $g(t)-g(t-1)$, fails to decrease. This method is like the second one, except that it is less sensitive (in terms of number of iterations) to random changes in image content (i.e. various genuine signal related differences). Figures B.2-B.5 show slices of a six bit image, a nine bit image, and a nine bit image after smoothing has been performed (5 and 10 cycles).

$g(t-1)$ = image at iteration t-1

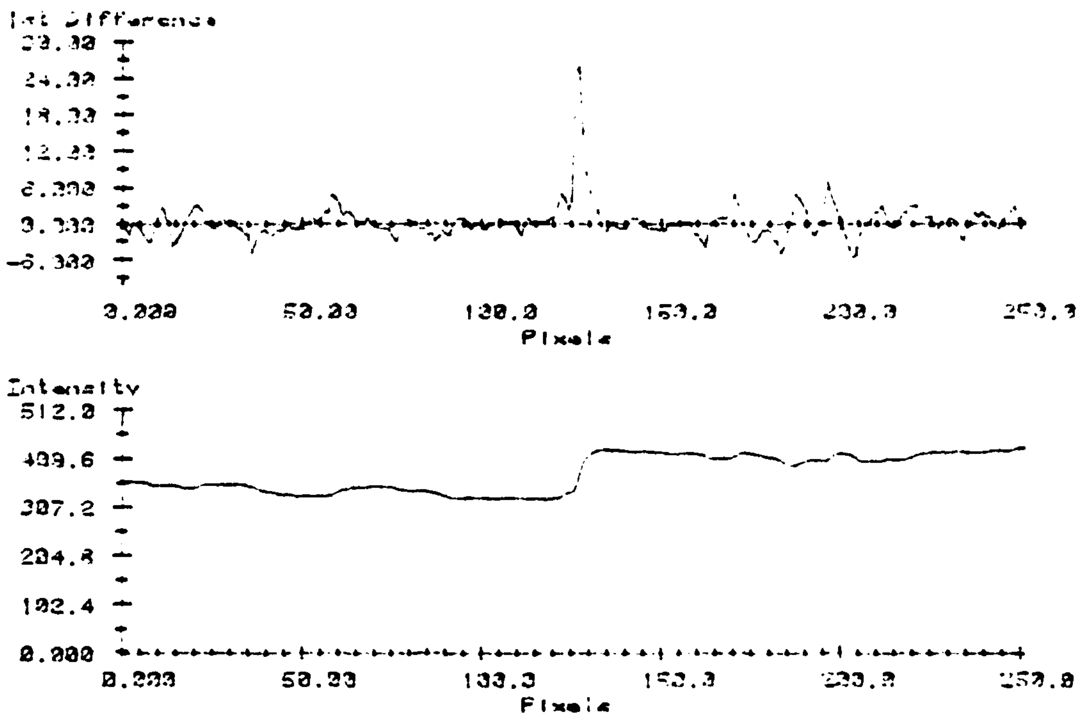
$g(t)$ = image at iteration t



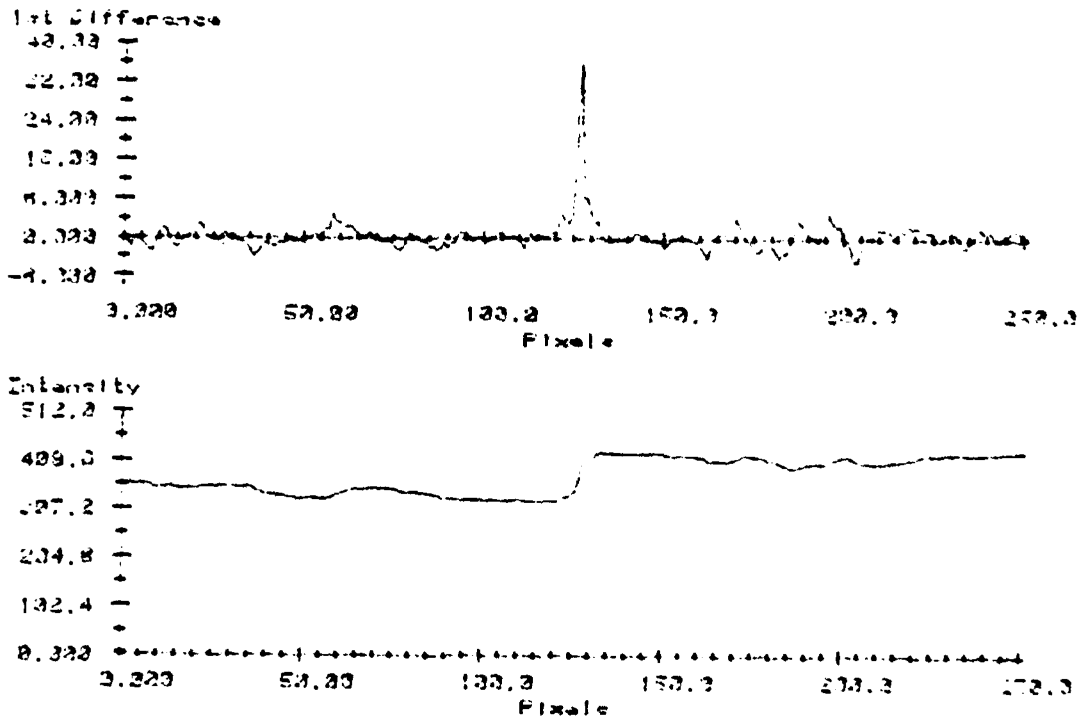
A Six Bit Image Slice
Figure B.2



A Nine Bit Image Slice
Figure B.3



A Nine Bit Image Slice After 5 Cycle Smoothing
Figure B.4



A Nine Bit Image Slice After 10 Cycle Smoothing
Figure B.5

Appendix C

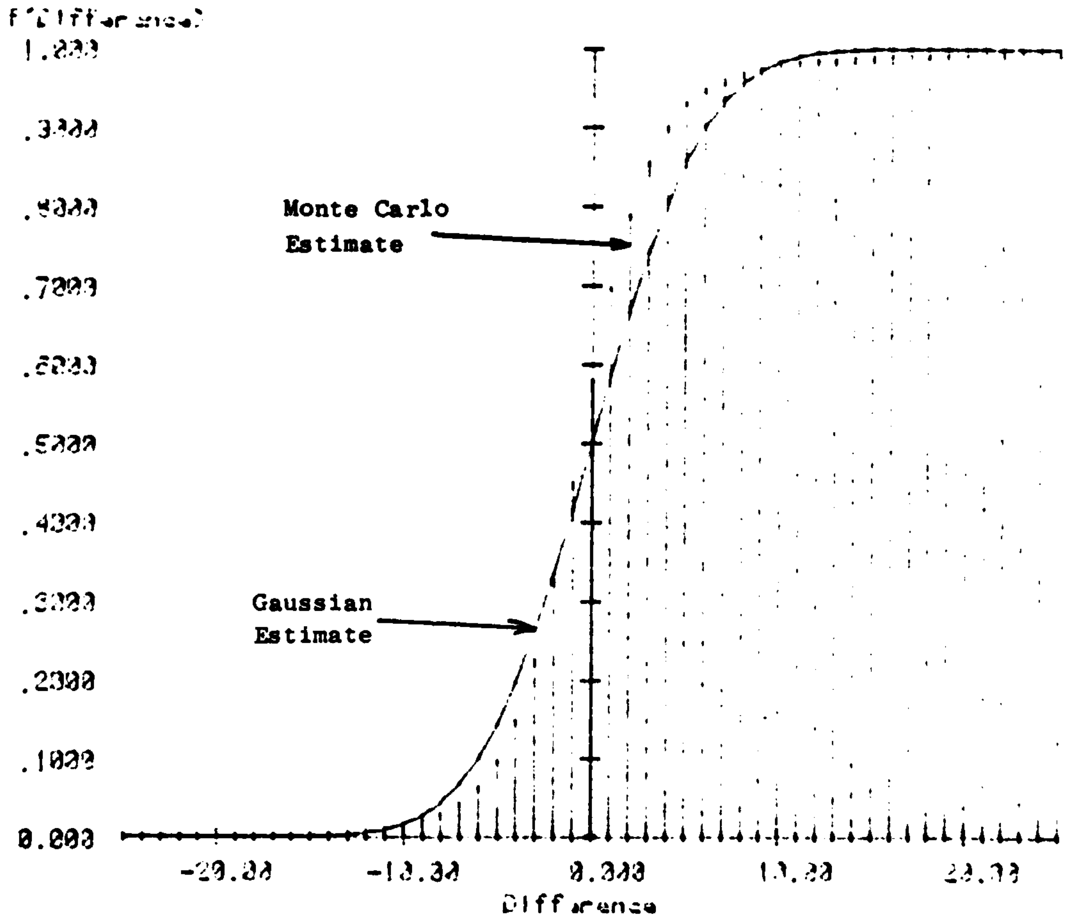
Selection of Thresholds in a Segmentation Process

Segmentation processes generally consist of clustering many measurements from similar objects together, or conversely separating a few highly dissimilar measurements from a large sea of similar ones. If we have a model for expected measurement error, we can form a function which relates any particular measurement value to the probability that that measurement represents a noise variation or a signal based variation. We then can select an acceptable threshold for separating "discontinuous" phenomenon from "continuous" phenomenon. (In discretized spaces each measurement may be considered discontinuous from the next. However, if a piecewise continuous function was sampled to form the discrete function, this definition of continuity is not useful. We wish a definition of discontinuity such that points at discontinuities in the underlying continuous space function also represent discontinuous points in the sampled function.)

First we need to define a difference variable. For edge detection/region growing this is simply the intensity difference computed from adjacent averaged intensity neighborhoods (the most primitive being single pixels). Possibly, different neighborhood orientations (such as vertical vs. horizontal differences) should be considered drawn from different populations (therefore handled as separate sets of data). For significant angle difference detection (corner detection), angle differences can be used. For depth correlation errors, depth differences from adjacent connect (in one or the other stereo image pairs) depth associated neighborhoods define a difference variable. A good difference variable is a zero mean random variable, for all measurements taken from a similar set of data.

If we assume many more similar readings (i.e. many more difference sites which are associated with no edge, than difference sites which are locations of edges), than dissimilar readings we may estimate the noise distribution functions using an one of several standard techniques (by assuming all the readings represent noise variation - the number of noise readings must be >> than the number of signal readings). For simple discrete cases we can use either a binomial model (good because there may be a position in the distribution where no noise will be detected as signal) or a normal model (good because it can be characterized easily by two parameters the mean μ and the variance σ). For more complex distribution types we may use a Monte Carlo method.

Armed with a typical noise distribution we may select a threshold which labels a particular measurement as a noise point with a known error probability p (Figure C.1). We need not threshold negative differences at the same significance level as positive ones, but we generally would expect to do so. This method for threshold selection is related to several goodness-of-fit methods commonly described in the literature [30].



Estimates of the Distribution of Single Adjacent Pixel Differences (An Example Difference Variable)
Figure C.1

Appendix D

Region and Vertex Label Sets

D.1. Labeling Algorithm

As a final object segmentation technique, we resort to a "Waltz-like" labeling scheme for assigning types to regions and half chunks. We are not interested in the intimate descriptions of line type that Waltz himself obtained, but rather want to separate the half chunks into three major types. These are lighting derived, true object derived, and occlusion derived. The first and the third types are grouped and marked so as to be ignored during the object recognition phase of processing. The meaning of lighting derived types is self-evident (we mean highlight or shadow related half chunks, when the half chunk does not also coincide with an object related boundary). Occlusion derived types (UNLINKED- types) are generated at what Waltz called "occluded" or "occluding" boundaries (Figure D.1). Regions or surfaces are labeled as shadow, highlight, ground, or object. Ground represents the largest background region (or group of regions). Generally this is the tabletop, or backdrop.

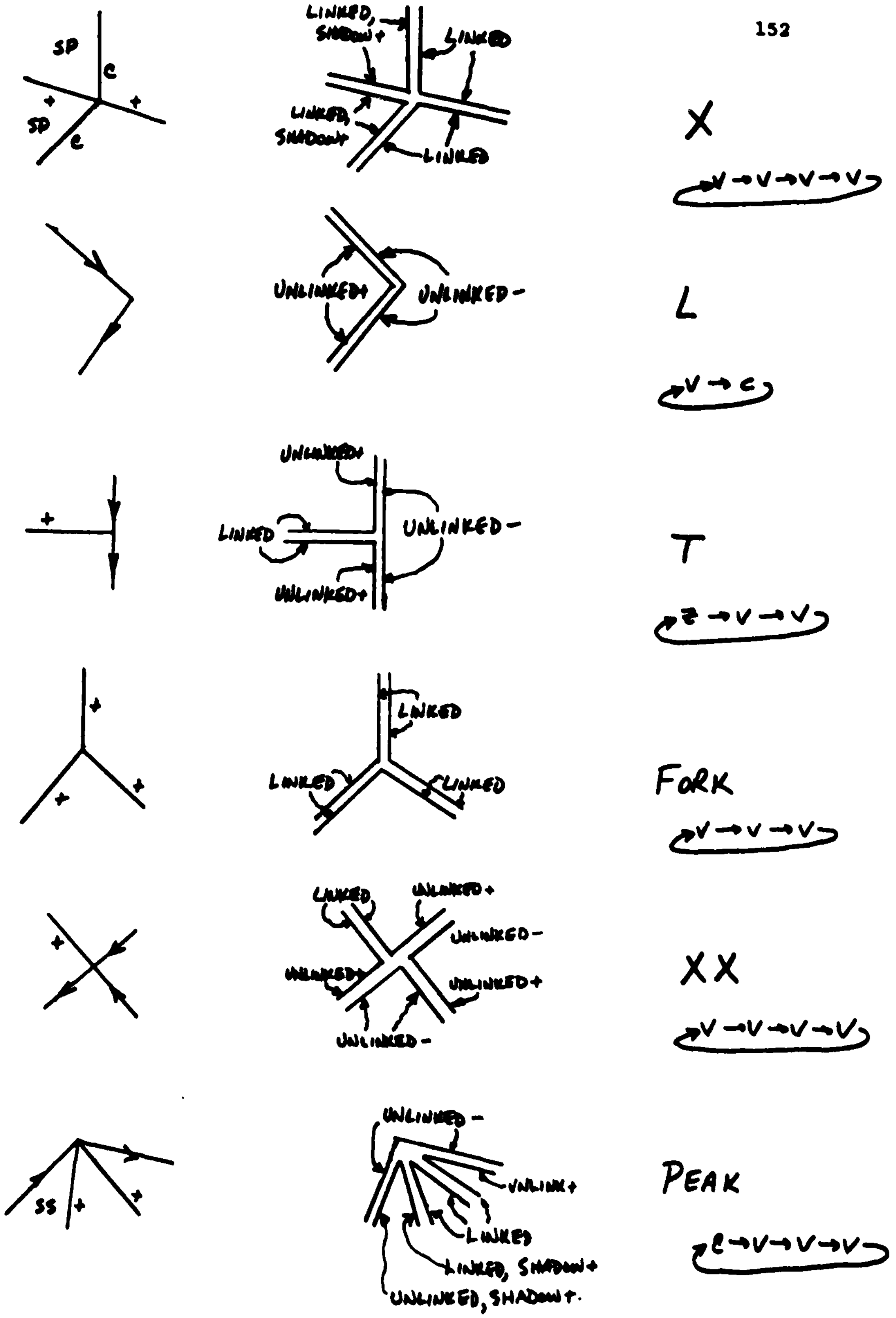


Figure D.1. Some Examples of Waltz Labelings and H-C Labelings

We begin our labeling algorithm on a H-C graph having each H-C assigned every possible label type. Iteratively we apply vertex-based constraints, then surface based constraints. All half chunks connect in X-link rings are processed jointly as vertices. All half chunks connect via R-links (alternatively, via E-link rings) are processed jointly as surfaces.

Each vertex label type applicable to a leg of a H-C is an "on" bit in the H-C type field for that leg (initially all are on). We scan through our vertex list for all matching vertices (same geometrical class, degree, and having compatible labelings), and accumulate the logical OR of these labelings (equivalent to cons'ing a list of all possible vertex matches). Possible labelings are ANDed with the initial H-C labelings. In the first iteration, this operation eliminates labelings incompatible with certain boundary feature properties.

Each E-link (connection along the boundary) is examined for adjacent H-C type compatibility. This causes the labelings on either side of the linkage to be the logical AND of the labelings prior to this processing. This operation is equivalent to computing the intersection of the the two label sets.

After the vertex-based processing (processing equivalent to Waltz's "filtering" algorithm), surface-based constraints are applied. Each surface aggregation is labeled as shadow, highlight, ground, or object, by matching indicator label types on all the H-C's encoding the surface. Then the labels which are incompatible with the region type are removed from each H-C.

These operations continue until a specified number of iterations are concluded, or until steady state is reached (generally the later). This labeling scheme is unique for three reasons. First, it is based on half chunks which are themselves unique. Second, it incorporates region constraints as well as boundary constraints. Finally, initial labelings are assigned and constrained by directly measured feature properties in addition to semantic types (the region and boundary labelings). In some sense the vertex geometrical type could be considered as one of these measured feature properties (convex, concave, straight).

D.2. Nomenclature

Boundary Segment (H-C Legs) Mnemonics:

LK	Linked
UP	Unlinked+
UM	Unlinked-
HP	Pure Highlight+
HM	Pure Highlight-
SP	Pure Shadow+
SM	Pure Shadow-
GR	Unlinked-, Ground
US	Unlinked-, Shadow+
UH	Unlinked+, Highlight+
MH	Unlinked-, Highlight+
MS	Unlinked+, Shadow+
LS	Linked, Shadow+
LH	Linked, Highlight+

Feature Property Mnemonics:

DIFF	Boundary Diffusion
LGTA	Absolute Lightness
LGTR	Relative Lightness
TANG	Tangency Difference
RAT	Ratio Variance
VARY	Side Variance
REFL	Reflective Variance
TEX	Roughness
REAL	Real/Subjective Index
DEEP	Depth Variance
MOVE	Motion Variance
EMBD	Region Embeddedness Index

Feature Property State Suffix:

#	Set to FALSE (no variation, not diffused, lightest, light>dark, most embedded)
&	Set to MAYBE (not determinable, object level embeddedness)
$	Set not-applicable-property
<none>	Set TRUE (variation, diffused, not lightest, dark>light, ground level embedding)

Half Chunk Angle Properties:

C	Concaved (> 180 deg.)
V	Convex (< 180 deg.)
Z	Zero, Straight (= 180 deg.)
?	C, V, or Z

Region Label Format:

(Indicator Label Concatenation ,
Compatibility Label Concatenation) [comments]

Vertex Label Format:

```
( ( H-C List 0 ) ( H-C List 1 ) ... ( H-C List n) )  [comments]
```

H-C List Format:

```
( H-C Angle Property , ( Leg List 0 ) , ( Leg List 1) )
```

Leg List Format:

```
( Feature Property Concatenation ,
    Boundary Segment labels Concatenation )
```

**Concatenations are mnemonics and suffixes separated by '+' (OR) or '-' (NOT(AND) - legal in label concatenations only)

D.3. Example Label Definition File

```
[INITIAL REGIONS - 2-JAN-79]

(LK+UP,LK+UP+UM+SM+HM)   [OBJECT]
(GR,GR)                  [GROUND]
(HP+UH+LH,LH+HP+UH+MH)   [PURE HIGHLIGHT]
(SP+MS+LS,MS+US+SP+LS)   [PURE SHADOW]
(,)

[INITIAL VERTICES - 2-JAN-79]

[2-DEG]

( (?,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (?,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK)) )

( (C,(DEEP#+MOVE#+DIFF#+EMBD#+LGTR#,LK),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTR#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,LH),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTA#+LGTR,LH)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#+LGTR#,LK),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTR#,LK))
    (Z,(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,LH),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTA#+LGTR,LH)) )

( (?,(DEEP#+MOVE#+DIFF#+EMBD#,LS),(DEEP#+MOVE#+DIFF#+EMBD#,LS))
    (?,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK)) )

( (?,(DEEP#+MOVE#+DIFF#+EMBD#,LS),(DEEP#+MOVE#+DIFF#+EMBD#,LS))
    (?,(DEEP#+MOVE#+DIFF#+EMBD#,LS),(DEEP#+MOVE#+DIFF#+EMBD#,LS)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,LH),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTA#+LGTR,LH))
    (Z,(DEEP#+MOVE#+DIFF#+EMBD#+LGTR#,LS),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTR#,LS)) )

( (V,(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,LH),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTA#+LGTR,LH))
    (C,(DEEP#+MOVE#+DIFF#+EMBD#+LGTR#,LS),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTR#,LS)) )

( (?,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
    (?,(EMBD,GR),(EMBD,GR)) )
( (?,(DEEP#+MOVE#+DIFF#+EMBD#,MS),(DEEP#+MOVE#+DIFF#+EMBD#,MS))
    (?,(EMBD,GR),(EMBD,GR)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,UH),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTA#+LGTR,UH))
    (Z,(EMBD+LGTR#,GR),(EMBD+LGTR#,GR)) )
```

```
( (V,(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,UH),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTA#+LGTR,UH))
   (C,(EMBD+LGTR#,GR),(EMBD+LGTR#,GR)) )

( (?,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
   (?,(DIFF#+EMBD#,UM),(DIFF#+EMBD#,UM)) )

( (?,(DEEP#+MOVE#+DIFF#+EMBD#+LGTR#,UP),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTR#,UP))
   (?,(DIFF#+EMBD#+LGTA#+LGTR,MH),(DIFF#+EMBD#+LGTA#+LGTR,MH)) )

( (?,(DEEP#+MOVE#+DIFF#+EMBD#,MS),(DEEP#+MOVE#+DIFF#+EMBD#,MS))
   (?,(DIFF#+EMBD#,US),(DIFF#+EMBD#,US)) )

( (?,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
   (?,(DIFF#+EMBD#,US),(DIFF#+EMBD#,US)) )

( (?,(DEEP#+MOVE#+DIFF#+EMBD#,MS),(DEEP#+MOVE#+DIFF#+EMBD#,MS))
   (?,(DIFF#+EMBD#,UM),(DIFF#+EMBD#,UM)) )

( (?,(DEEP#+MOVE#+DIFF#+EMBD#+LGTR#,MS),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTR#,MS))
   (?,(DIFF#+EMBD#+LGTA#+LGTR,MH),(DIFF#+EMBD#+LGTA#+LGTR,MH)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,UH),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTA#+LGTR,UH))
   (Z,(DIFF#+EMBD#+LGTR#,UM),(DIFF#+EMBD#+LGTR#,UM)) )

( (V,(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,UH),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTA#+LGTR,UH))
   (C,(DIFF#+EMBD#+LGTR#,UM),(DIFF#+EMBD#+LGTR#,UM)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#,UH),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTA#,UH))
   (Z,(DIFF#+EMBD#+LGTA#,MH),(DIFF#+EMBD#+LGTA#,MH)) )

( (V,(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#,UH),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTA#,UH))
   (C,(DIFF#+EMBD#+LGTA#,MH),(DIFF#+EMBD#+LGTA#,MH)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,UH),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTA#+LGTR,UH))
   (Z,(DIFF#+EMBD#+LGTR#,US),(DIFF#+EMBD#+LGTR#,US)) )

( (V,(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,UH),(DEEP#+MOVE#+DIFF#+EMBD#+
        LGTA#+LGTR,UH))
   (C,(DIFF#+EMBD#+LGTR#,US),(DIFF#+EMBD#+LGTR#,US)) )

( (?,(EMBD,GR),(EMBD,GR))
   (?,(EMBD,GR),(EMBD,GR)) )

( (Z,(DIFF+LGTA#+LGTR+EMBD#,HP),(DIFF+LGTA#+LGTR+EMBD#,HP))
   (Z,(DIFF+LGTR#,HM),(DIFF+LGTR#,HM)) )

( (V,(DIFF+LGTA#+LGTR+EMBD#,HP),(DIFF+LGTA#+LGTR+EMBD#,HP))
   (C,(DIFF+LGTR#,HM),(DIFF+LGTR#,HM)) )

( (?,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP))
   (?,(DIFF+LGTR+DEEP#+EMBD#,SM),(DIFF+LGTR+DEEP#+EMBD#,SM)) )

( (?,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP))
   (?,(EMBD,GR),(EMBD,GR)) )
```

```
[3-DEG]

[?-JUNCTIONS]

( (?,(EMBD,GR),(EMBD,GR))
  (?,(EMBD,GR),(EMBD,GR))
  (?,(EMBD,GR),(EMBD,GR)) )

[Y-JUNCTIONS]

( (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK)) )

( (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(EMBD,GR),(EMBD,GR))
  (V,(EMBD,GR),(EMBD,GR)) )

( (V,(EMBD,GR),(EMBD,GR))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(DIFF#+EMBD#,US),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP)) )

( (V,(EMBD,GR),(EMBD,GR))
  (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF#+EMBD#,US))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (V,(EMBD,GR),(EMBD,GR))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

[W-JUNCTIONS]

( (C,(DIFF#+EMBD#,UM),(DIFF#+EMBD#,UM))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (C,(EMBD,GR),(EMBD,GR))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

[T-JUNCTIONS]

( (Z,(EMBD,GR),(EMBD,GR))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (Z,(EMBD,GR),(EMBD,GR))
  (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF#+EMBD#,US))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (Z,(EMBD,GR),(EMBD,GR))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(DIFF#+EMBD#,US),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP)) )

( (Z,(EMBD,GR),(EMBD,GR))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,MS),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP))
  (V,(DIFF+LGTR+DEEP#+EMBD#,SM),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (Z,(EMBD,GR),(EMBD,GR))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DIFF+LGTR+DEEP#+EMBD#,SM))
  (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DEEP#+MOVE#+DIFF#+EMBD#,MS)) )

( (Z,(EMBD,GR),(EMBD+LGTR#,GR))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,UH),(DIFF+LGTA#+LGTR+EMBD#,HP))
  (V,(DIFF+LGTR#,HM),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )
```

```
( (Z,(EMBD+LGTR#,GR),(EMBD,GR))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DIFF+LGTR#,HM))
  (V,(DIFF+LGTA#+LGTR+EMBD#,HP),(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,UH)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#+LGTR#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,LH),(DIFF+LGTA#+LGTR+EMBD#,HP))
  (V,(DIFF+LGTR#,HM),(DEEP#+MOVE#+DIFF#+EMBD#,LK)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#+LGTR#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DIFF+LGTR#,HM))
  (V,(DIFF+LGTA#+LGTR+EMBD#,HP),(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,LH)) )

( (Z,(DIFF#+EMBD#,UM),(DIFF#+EMBD#+LGTR#,UM))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,UH),(DIFF+LGTA#+LGTR+EMBD#,HP))
  (V,(DIFF+LGTR#,HM),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (Z,(DIFF#+EMBD#+LGTR#,UM),(DIFF#+EMBD#,UM))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DIFF+LGTR#,HM))
  (V,(DIFF+LGTA#+LGTR+EMBD#,HP),(DEEP#+MOVE#+DIFF#+EMBD#+LGTA#+LGTR,UH)) )

( (Z,(DIFF#+EMBD#,UM),(DIFF#+EMBD#,UM))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(DIFF#+EMBD#,UM),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DIFF#+EMBD#,UM)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(EMBD,GR),(EMBD,GR))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DIFF#+EMBD#,UM)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(DIFF#+EMBD#,UM),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(EMBD,GR),(EMBD,GR)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(EMBD,GR),(EMBD,GR))
  (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF#+EMBD#,US)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(DIFF#+EMBD#,US),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP))
  (V,(EMBD,GR),(EMBD,GR)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#+LGTR#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(DIFF#+EMBD#,UM),(DIFF+LGTR#,HM))
  (V,(DIFF+LGTA#+LGTR+EMBD#,HP),(DIFF#+EMBD#+LGTA#+LGTR,MH)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#+LGTR#,UP))
  (V,(DIFF#+EMBD#+LGTA#+LGTR,MH),(DIFF+LGTA#+LGTR+EMBD#,HP))
  (V,(DIFF+LGTR#,HM),(DIFF#+EMBD#,UM)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(DIFF#+EMBD#,UM),(DIFF+LGTR+DEEP#+EMBD#,SM))
  (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF#+EMBD#,US)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(DIFF#+EMBD#,US),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP))
  (V,(DIFF+LGTR+DEEP#+EMBD#,SM),(DIFF#+EMBD#,UM)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(DIFF#+EMBD#,UM),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(DIFF#+EMBD#,UM),(DIFF#+EMBD#,UM)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(DIFF#+EMBD#,UM),(DIFF#+EMBD#,UM))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DIFF#+EMBD#,UM)) )
```

```
( (Z,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(EMBD,GR),(EMBD,GR))
  (V,(EMBD,GR),(EMBD,GR)) )

[4-DEG]

[?-JUNCTIONS]

( (?,(EMBD,GR),(EMBD,GR))
  (?,(EMBD,GR),(EMBD,GR))
  (?,(EMBD,GR),(EMBD,GR))
  (?,(EMBD,GR),(EMBD,GR)) )

[POINT-JUNCTIONS]

( (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK)) )

( (V,(EMBD,GR),(EMBD,GR))
  (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF#+EMBD#,US))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,MS),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP))
  (V,(DIFF+LGTR+DEEP#+EMBD#,SM),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (V,(EMBD,GR),(EMBD,GR))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DIFF+LGTR+DEEP#+EMBD#,SM))
  (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DEEP#+MOVE#+DIFF#+EMBD#,MS))
  (V,(DIFF#+EMBD#,US),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP)) )

( (V,(DIFF#+EMBD#,UM),(DIFF+LGTR+DEEP#+EMBD#,SM))
  (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF#+EMBD#,US))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,MS),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP))
  (V,(DIFF+LGTR+DEEP#+EMBD#,SM),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (V,(DIFF+LGTR+DEEP#+EMBD#,SM),(DIFF#+EMBD#,UM))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DIFF+LGTR+DEEP#+EMBD#,SM))
  (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DEEP#+MOVE#+DIFF#+EMBD#,MS))
  (V,(DIFF#+EMBD#,US),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP)) )

[PENTA-JUNCTIONS]

( (C,(EMBD,GR),(EMBD,GR))
  (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF#+EMBD#,US))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (C,(EMBD,GR),(EMBD,GR))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(DIFF#+EMBD#,US),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP)) )

( (C,(DIFF#+EMBD#,UM),(DIFF+LGTR+DEEP#+EMBD#,SM))
  (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF#+EMBD#,US))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (C,(DIFF+LGTR+DEEP#+EMBD#,SM),(DIFF#+EMBD#,UM))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
  (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
  (V,(DIFF#+EMBD#,US),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP)) )
```

```
( (C,(EMBD,GR),(EMBD,GR))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (C,(DIFF#+EMBD#,UM),(DIFF#+EMBD#,UM))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (Z,(EMBD,GR),(EMBD,GR))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (Z,(DIFF#+EMBD#,UM),(DIFF#+EMBD#,UM))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

[DEG-5]

[?-JUNCTIONS]

( (?,(EMBD,GR),(EMBD,GR))
   (?,(EMBD,GR),(EMBD,GR))
   (?,(EMBD,GR),(EMBD,GR))
   (?,(EMBD,GR),(EMBD,GR))
   (?,(EMBD,GR),(EMBD,GR)) )

[POINT-JUNCTIONS]

( (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK)) )

( (Z,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK)) )

( (V,(EMBD,GR),(EMBD,GR))
   (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF#+EMBD#,US))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LS),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP))
   (V,(DIFF+LGTR+DEEP#+EMBD#,SM),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (V,(EMBD,GR),(EMBD,GR))
   (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF#+EMBD#,US))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,MS),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP))
   (V,(DIFF+LGTR+DEEP#+EMBD#,SM),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (V,(EMBD,GR),(EMBD,GR))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DIFF+LGTR+DEEP#+EMBD#,SM))
   (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DEEP#+MOVE#+DIFF#+EMBD#,MS))
   (V,(DIFF#+EMBD#,US),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP)) )

( (V,(EMBD,GR),(EMBD,GR))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DIFF+LGTR+DEEP#+EMBD#,SM))
   (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DEEP#+MOVE#+DIFF#+EMBD#,LS))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
   (V,(DIFF#+EMBD#,US),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP)) )
```

```
( (V,(DIFF#+EMBD#,UM),(DIFF+LGTR+DEEP#+EMBD#,SM))
    (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF#+EMBD#,US))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LS),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP))
    (V,(DIFF+LGTR+DEEP#+EMBD#,SM),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (V,(DIFF#+EMBD#,UM),(DIFF+LGTR+DEEP#+EMBD#,SM))
    (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF#+EMBD#,US))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,MS),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP))
    (V,(DIFF+LGTR+DEEP#+EMBD#,SM),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (V,(DIFF+LGTR+DEEP#+EMBD#,SM),(DIFF#+EMBD#,UM))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DIFF+LGTR+DEEP#+EMBD#,SM))
    (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DEEP#+MOVE#+DIFF#+EMBD#,MS))
    (V,(DIFF#+EMBD#,US),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP)) )

( (V,(DIFF+LGTR+DEEP#+EMBD#,SM),(DIFF#+EMBD#,UM))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DIFF+LGTR+DEEP#+EMBD#,SM))
    (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DEEP#+MOVE#+DIFF#+EMBD#,LS))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
    (V,(DIFF#+EMBD#,US),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP)) )

[PENTA-JUNCTIONS]

( (C,(EMBD,GR),(EMBD,GR))
    (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF#+EMBD#,US))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (C,(EMBD,GR),(EMBD,GR))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
    (V,(DIFF#+EMBD#,US),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP)) )

( (C,(DIFF#+EMBD#,UM),(DIFF+LGTR+DEEP#+EMBD#,SM))
    (V,(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP),(DIFF#+EMBD#,US))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (C,(DIFF+LGTR+DEEP#+EMBD#,SM),(DIFF#+EMBD#,UM))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP))
    (V,(DIFF#+EMBD#,US),(DIFF+LGTA+LGTR#+DEEP#+EMBD#,SP)) )

( (C,(EMBD,GR),(EMBD,GR))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (C,(DIFF#+EMBD#,UM),(DIFF#+EMBD#,UM))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

( (Z,(EMBD,GR),(EMBD,GR))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
    (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )
```

```
( (Z,(DIFF#+EMBD#,UM),(DIFF#+EMBD#,UM))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,UP),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,LK))
   (V,(DEEP#+MOVE#+DIFF#+EMBD#,LK),(DEEP#+MOVE#+DIFF#+EMBD#,UP)) )

[END-OF-VERTICES]
```

Appendix E

Image Formats

E.1. General Image Format

Images of any resolution and pixel sizes from 1 bit to 36 bits may be encoded. Optional user information may be encoded. Several such option fields commonly used are included in this description. Programs are available for digitizing images into 252x238x6 bit, 252x238x9 bit, and 504x476x6 bit forms.

Field(word)	Description
0	Picture Length (Image+Header)
1-3	UP, RIGHT, LEFT Links/Markers (See Color Triples)
4	Atom Type (Always = 2)
5	Highest X Pixel Coordinate (X Dimension Minus 1) - High X
6	Highest Y Pixel Coordinate (Y Dimension Minus 1) - High Y
7	Number of Gray levels (Quantization Steps - for 6 bit images = 64 steps) - GL
8	Minimum Pixel Value - Min I
9	Maximum Pixel Value - Max I
10	Average Pixel Value - Ave I
11	Offset to Pixel Pointer List
12	Offset to Picture Data
13-15	Offsets to Interleaved Picture Data (Only for 504x476x6 bit pictures directly from the imaging routines)
16-(15+m)	User's Fields (m is words of user's fields)

```
     Commonly Computed Fields
     -------- -------- ------

Pixel Size = ( C( C( O(11) ) ) .and. 007700000000 )/100000000

User Fields Length =  C( O(11) ) - O(16)

Offset to First Picture Word (Pixel [High Y,0]) = C( O(12) )

Offset to Pixel X,Y = Offset to First Picture Word + C( O(11) +
                      ( X*(Y - High Y) .mod. (36/Pixel Size) ) ) +
                      ( X*(Y - High Y) / (36/Pixel Size) )

n .and. m = Logical "AND" of n and m;  O(n) = Offset n;
n .mod. m = Remainder of n/m;          C(n) = Contents of n;
```

Common User Defined Fields Description
------ ---- ------- ------ -----------

16	Time (in Milliseconds) Since Last Image Was Taken (for picture sequences)
17	Edge Speeds (See Edge Pictures) - E Speed
18	Edge Position Modifier Field size (See Edge Pictures) - E Pos

E.2. Color Image Triples

Color images are stored as sequences of images consisting of a black/white image followed by one or more single color images (red, green, or blue spectra). All images are in the general image format. The red, green, or blue images may be omitted. The red image is associated with the UP link in the black/white image. The green image is associated with RIGHT, and the blue with LEFT. If a link field of the black/white image is non-zero, the associated colored image is available for reading.

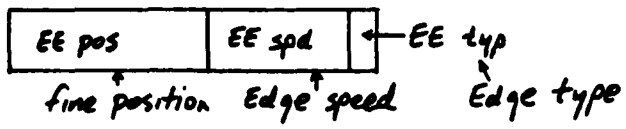
E.3. Image Format for Edges

The output of all the edge detection systems described in this thesis are in the form of image triples. The first image is the input gray scale image (dimensioned (High X + 1) by (High Y + 1)). The next is a Y Edge image having one pixel for each junction of pairs of pixels in the gray scale image lying on a vertical line. The dimension of this image is (High X + 1) by (High Y). Following is the X Edge image, having one pixel for each junction of pairs of pixels in the gray scale image lying on a horizontal line. This image is dimensioned (High X) by

(High Y + 1).

Each non-zero pixel in an edge picture represents a significant intensity transition with maximum tendency between the corresponding gray scale image pixels. The edge orientation is horizontal (in the Y Edge image) or vertical (in the X Edge image). Edge pixels carry edge type, transition size, and fine positioning fields. Edge type fields require no bits or one bit. Transition size field requires the number bits needed to encode the maximum edge speed (see Edge Speed field in the general image format section). The edge speed is an index to the edge mask that matched the local intensity phenomenon. This nominally requires numbers from 1 to 6. The basic position of an edge is specified by its location in the picture matrices, however, for fine tuning position plus or minus 0.5 pixel positions the edge position modifier field is used (EE pos). The size of this field is variable (see Edge Position Modifier field in the general image format section).

Edge Pixel Schematic



Edge Position Fine Tuning
---- -------- ---- ------

E pospos = 2**E pos - 1

Position For X Edge At X,Y = (X + (EE pos - E pospos/2)/E pospos,Y)

Position For Y Edge At X,Y = (X,Y + (EE pos - E pospos/2)/E pospos)

E.4. Image Format for Regions

The data sets produced after region aggregation consist of image quadruples. The first is an intensity image, the second is a region index image (same dimension as the intensity image), and the last two are X and Y edge images (formatted as in edge data sets). Each pixel in the region image contains a region marking number. This number indexes into a region list. Each region has one region list entry, containing the following fields:

Field(word)	Description
0	Area (pixels)
1	X Center of Mass
2	Y Center of Mass
3	Average Intensity
4	X for a Pixel in the Region
5	Y for a Pixel in the Region
6	Formation Order
7	X Edge Density
8	Y Edge Density
9	Perimeter
10	Standard Deviation of Intensity
11	Perimeter with Frame of Picture

The region list is written as a binary list formatted as follows:

Words	Description
1	High X for Original Image
1	High Y for Original Image
1	Gray Levels for Original Image
1	Time Original Image Was Digitized
1	Region List Atoms
i*12	Words of Region List Atoms

E.5. V-S And V-S-S Format

Vertex-String[-Surface] (V-S[-S]) format graphs are written in binary to the filing system as lists of atoms and links:

Words	Description
1	High X for Original Image
1	High Y for Original Image
1	Gray Levels for Original Image
1	Time (Optional... In V-S-S, not V-S)
1	String Atoms (m)
m*10	Words of String Atoms
1	Edge Point Atoms (n)
n*4	Words of Edge Point Atoms
1	Vertex Atoms (l)
l*4	Words of Vertex Atoms
1	Vertex-String Associations (j)
j	Words of Vertex-String Associations

Optional... in V-S-S format:

1	Region Atoms (i)
i*12	Words of Region Atoms (See Region List Format for Each Entry)

String Atoms:

Field(word)	Description
0	Edges<18,18>, Real Edges<0,18>
1	R0 (left)<18,18>, R1 (right)<0,18>
2	Average Gray Level in R0
3	Average Gray Level in R1
4	Average (Gray Level)**2 in R0
5	Average (Gray Level)**2 in R1
6	Average (Gray Level R0 - Gray Level R1)**2
7	Average Boundary Diffusion
8	Low Vertex<18,18>, High Vertex<0,18>
9	Low Edge Index<18,18>, High Edge Index<0,18>

**<n,m> notation: n is field position (bits right of the field)
m is field size (in bits)

Edge Atoms:

Field(word)	Description
0	Subjective/Objective<35,1>, Transition Size<28,7>, Edge Index<14,14>, String Index<0,14>
1	X Position
2	Y Position
3	Tangent Angle (Radians)

Vertex Atoms:

Field(word)	Description
0	X Position
1	Y Position
2	Position Uncertainty
3	Psuedo/Real<35,1>, Vertex-String Association Low<18,17>, Vertex-String Association High<0,18>

Vertex-String Associations:

Field(word)	Description
0	Vertex Index<18,18>, String Index<0,17>

S.6. H-C Format

The Half Chunk Graph format is used internally to the multi-image correlation system, and is written to provide input to the object recognition system. When written, the format is as follows:

Words	Description
1	Time Elapsed Since Last Packet
1	Total Time Elapsed
1	Frame Intervals in This Packet
1	Objects Detected (Connected H-C Graphs After Object Segmentation) - m
m*8	Words of Object Atoms
1	Number of Regions/Surfaces - n
n	Words of Object-Surface Associations
1	Number of H-C Atoms - i
i*7	Words of H-C Atoms
1	Number of XYZ Atoms - i (Same as H-C's)
i*4	Words of XYZ Atoms
1	Number of H-C Associations - j=i*4
j	Words of H-C Associations/Links: E-link, Side 0 E-link, Side 1 X-link, Side 0 X-link, Side 1

Object Atom Format:

Field(word)	Description
0	Object Type Flags, Object Is 2-d<35,1>, Object Type<33,2>, Type# Type Description 0 Significant Object 1 Insignificant Object 2 Lighting Effect Object 3 Ground (Occluded) Object Transform Type<30,2>, Type# Type Description 0 3-d Translation, 3-d Rotation 1 3-d Translation, 2-d Rotation 2 3-d Translation Only 3 No Transform Object Has Curved H-C's<32,1>
1	Low H-C Index<18,18>, High H-C Index<0,18>
2-4	X, Y, Z Translations
5-7	Alpha, Beta, Gamma Rotation Angles

Half Chunk Atom Format (H-C):

Field(word)	Description
0	Type Flags<18,18>, 2D/3D, Curved/Vertex, Object/Non-object/Lighting Surface Association<0,18>
1	Tangent/Change in Tangent
2	Length Side 0
3	Length Side 1
4	Properties Side 0
5	Properties Side 1

Prop#	Prop Description
0-3	Intensity Chops
4	Boundary Diffusion
5	Absolute Lightness (Darkest, Lightest)
6	Relative Lightness (Dark/Light, Light/Dark)
7	Tangency Difference
8	Ratio Variance
9	Side Variance
10	Reflective Variance
11	Roughness
12	Real/Subjective Index
13	Depth Variance
14	Motion Variance
15	Region Embeddedness Index

Field(word)	Description
6	Labeling Side 0<18,18>, Labeling Side 1<0,18> Linked, Unlinked+, Unlinked-, Pure Highlight+, Pure Highlight-, Pure Shadow+, Pure Shadow-, Unlinked- & Ground, Unlinked+ & Shadow+, Unlinked- & Shadow+, Unlinked+ & Highlight+, Unlinked- & Highlight+, Linked & Shadow+, Linked & Highlight+

Optional... Not in H-C Format Written to Filing System:

7	X Center
8	Y Center
9	X Side 0
10	Y Side 0
11	X Side 1
12	Y Side 1
13	Spare Word

XYZ Atom Format:

Field(word)	Description
0	Curved/Vertex<35,1>, Associated H-C Index Number<18,17>, 2D/3D<17,1>, Frame Number<0,17> (Not on Written XYZ)
1	X Coordinate for H-C Center
2	Y Coordinate for H-C Center
3	Z Coordinate for H-C Center (0 for 2D)

Object-Surface Association:

Field(word)	Description
0	Surface/Region Number<18,18>, Object Graph Number<0,17>

E.7. Object Library Format

Streams of object packets are read by the recognition program, one isolated subgraph at a time, to be incorporated into the object library. This is done by either matching an existing object library entry or by forming a new one (if no initial object library file exists, the first object matches nothing and is used to begin a library). Library files are made up of a components list, followed by a stream of object graphs (expressed in terms of references to the atoms in the components list):

Words	Description
1	Number of H-C Prototypes in the Components List - m
m*2	Words of H-C Prototype Atoms
1	Number of Object Graphs - n
n*20	Words of Object Graph Directory Entries
?	Object Graphs, Each Start on a New Block Boundary

Object Graph Format:

Words	Description
1	Number of H-C Component Indexes - m
m	Words of H-C Component Indexes
m*5	H-C Links: E-link, Side 0 E-link, Side 1 X-link, Side 0 X-link, Side 1 R-link

H-C Component Format:

Field(word)	Description
0	Non-shape Properties, Side 0<18,8>, Non-shape Properties, Side 1<27,8>, (selected from H-C Properties)
1	Length Ratio Property<0,18> Curvature Angle (Change of Tangent) Property<18,18>, Index into Components List<0,18>

Object Graph Directory Entry Format:

Field(word)	Description
0	Object Graph First Block<0,18>, Extension Graph First Block<18,18>
1	Blocks in Graph<0,18>, Time Entry Was Made<18,18>
2	Pointer To first Order Histogram (In Memory Only)<0,18>, Histogram Offset (In Memory Only)<18,18>,
3	Index To First Object Graph<0,18> (In Extension Graph Entries),
4-19	Object Name

Appendix F

Programs and Functions

F.1. Required by All:

LIBVIS.REL	Vision and Graphics Runtime Library
FASLIB.REL	Picture Digitizing Runtime Library
VISION.BEG	Vision and Graphics REQ'd Files
TTYIO.BEG	Terminal Input/Output
NUMIO.BEG	Numerical Input/Output
FILIO.BEG	File Input/Output
MISC.BEG	Keyword Processing
CALLI.BEG	System Calls
MAIN.BEG	Prologs and Epilogs
FLOAT.BEG	Floating Point Conversions
PICTUR.BEG	Picture Format Definitions
STRUCT.BEG	Array Definitions

F.2. Main Line System

Program(.lang)	Requires	Description
PSNAP(.BLISS)		Frame Integrate 8 Images
		Reads From Silicon Vidicon
		Writes To a Picture File
SM(.BLISS)		Smooth-Enhance
		Reads and Writes Picture Files
EDGES(.BLISS)	EP.BEG	Detect Variable Transition Edges
	E.BEG	-Multiple Template Operator
		Reads From a Picture File
		Writes Edge Triple File
FILTER(.BLISS)	EP.BEG	Remove Isolated Edges
		Reads and Writes Edge Triple Files
REGION(.BLISS)	AREA.BEG	Detect Connected Regions
	PI.BEG	-Multiple Homogeneity Operators
	EP.BEG	Reads Edge Triple Files
		Writes Region Quad Files
		Writes Region Lists Files
ELIST(.BLISS)	EP.BEG	Make Edge Lists
	SQ.BEG	Reads Region Quad Files
	PI.BEG	Writes Edge Lists Files
	STRING.BEG	
STRING(.BLISS)	SQ.BEG	Make V-S Graphs
	PI.BEG	Reads Edge Lists Files
	COMPAR.BEG	Writes V-S Files
	STRING.BEG	
REGSTR(.BLISS)	STATS.BEG	Make V-S-S Graphs
	SQ.BEG	Reads V-S Files
	PI.BEG	Reads Region Lists Files
	STRING.BEG	Writes V-S-S Files
	AREA.BEG	Writes Statistical Summary Files
SMLSTR(.BLISS)	SQ.BEG	Resegment V-S-S Graphs
	PI.BEG	Reads and Writes V-S-S Files
	STRING.BEG	
	AREA.BEG	
CORLAB(.BLISS)	CORLAB.BEG	Convert V-S-S Graphs to H-C Graphs
	SQ.BEG	Correlate H-C Graphs for Depth
	STRING.BEG	Correlate H-C Graphs for Motion
	AREA.BEG	Label H-C Graphs and Form Object
	PI.BEG	Graphs from Isolated Subgraphs
	STATS.BEG	Reads V-S-S Graphs
	HC.BEG	Reads Parameter Files
	CORLAB.CCL	Reads Vertex/Region Labels Files
		Writes H-C Files
GHIST(.BLISS)	PI.BEG	Label Objects, Form Object Libraries
	SQ.BEG	Reads H-C Files
	HC.BEG	Reads Object Library Files
	CL.BEG	Writes Object Library Files
	GHIST.CCL	

F.3. Other Programs

Program(.lang)	Description
CHEESE(.BLISS)	Take 252x238x6 Bit or 504x476x6 Bit Image Sequences -Color or B/W
FREEZE(.BLISS)	Take 252x238x6 Bit Motion Sequence
SM2(.BLISS)	Smooth-Enhance Images
EXSQ(.BLISS)	Expand-Squeeze Resolution
PIXS(.BLISS)	Display Gray Scale/Color Images
BPIXS(.BLISS)	Display Binary Pictures
D(.BLISS)	Display Binary Pictures On Plasma Panel
EPLOT(.BLISS)	Display Edge Detection Plots
EEXAM(.BLISS)	Display Edge Triples and Region Quad Files
RAMR(.BLISS)	Display in Color Edge Triples and Region Quad Files
EELIST(.BLISS)	Display Edge Lists Files
PSTR(.BLISS)	Display Region Lists, V-S, and V-S-S Files
DISPO(.BLISS)	Display H-C Files (DISPO.CCL)
SEELIB(.BLISS)	Print Object Library Directory Information
MEDGES(.BLISS)	Edge Detection - Adaptive Threshold Method
ER(.BLISS)	Edge Detection - Rosenfeld, Yakimovsky, Linear Difference
EW(.BLISS)	Edge Detection - Hueckel, Psuedo-Hueckel
ARITH(.BLISS)	Picture Arithmetic
ES(.BLISS)	Merge Edge Files For Color-Texture Superposition
MM(.BLISS)	Make Textured Regions Mask
SMILEX(.MACRO)	Expand-Reduce Image Resolution/Camera System
PRINTX(.MACRO)	Expand-Reduce Image Resolution/Files
EPARM(.BLISS)	Compute Edge Detector Parameters
ETUNE(.BLISS)	Plot Edge Detector Tuning Plots
EDX(.BLISS)	Plot Edge Detector Tuning Plots
SEDX(.BLISS)	Plot Edge Detector Tuning Plots
FRAMED(.BLISS)	Select a Window from an Image
PHIST(.BLISS)	Plot Histograms, Slices from Image Files
3DHIST(.BLISS)	Plot Image as a 3-D Function
PICNOS(.BLISS)	Estimate Image Noise Statistics
EM(.BLISS)	Edge Texture Image Generator
CTOM(.BLISS)	V-S-S Binary Format To ASCII List Format
MTOC(.BLISS)	V-S-S ASCII List Format to Binary Format
MREGS(.BLISS)	Merge Region List Entries
RSET(.BLISS)	Make Artificial Images
RCVT(.BLISS)	Scale Colors For Image Display
RVSPAC(.BLISS)	Edit V-S-S Sequences
PSPACE(.BLISS)	Edit Edge Triples Sequences
M(.BLISS)	Edit Image Sequences

VITA

Charles Jeremiah Jacobus was born November 4, 1951 in Chicago, Illinois. He attended Hinsdale Township High School Central in Hinsdale, Illinois and graduated in 1969. He attended the University of Illinois in Urbana-Champaign, and graduated with high honors in 1973 with Bachelor of Science in Electrical Engineering. He continued at the University of Illinois in graduate school and worked at the Coordinated Science Laboratory as a research assistant in the Advanced Automation Research Group. He received a Master of Science in Electrical Engineering in 1975 and continued on for a Doctorate in Electrical Engineering, received in 1979.